Knowledge management technology for integrated decision support systems in process industries

Knowledge management technology for integrated decision support systems in process industries

Edrisi Muñoz Mata

M.Sc. in Industrial Engineering

A Thesis presented for the degree of
Doctor of Philosophy

Directed by:
Prof. Dr. Antonio Espuña
Prof. Dr. Luis Puigjaner

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
Universitat Politècnica de Catalunya

December 2011

A mi familia con todo mi cariño

He aquí

una prueba para verificar

si tu misión en la tierra

ha concluido:

Si estas vivo

no ha concluido

Ilusiones Richard Bach (1977)

Summary

Nowadays, factors such as globalization of trade, market uncertainty and fierce competition involve dwindling error margins in enterprises. Two key aspects for achieve it are the viability and the competitiveness of enterprises, which highly depend on the effectiveness for taking their decisions related to their manufacturing characteristics, such as economic efficiency, product quality, flexibility or reliability. For this reason, companies have taken the task of developing better management information systems in order to help the decision makers to exploit data and models, enriching and improving decision-making.

In this sense, decision support systems must be improved in order to deal with the large amount of available data and the heterogeneity of existing modeling approaches available along the hierarchical levels in the enterprise structure. Hence, this thesis proposes the application of ontologies as a decision support tool. Ontologies are increasingly seen as a key semantic technology for addressing the problems created by data heterogeneities thus enabling data mining driving by knowledge processing.

The aim of this thesis is to contribute to the development of decision support tools for the process industry. As a decision support tool, it must be capable of becoming a robust model which interacts among the different decision hierarchical levels, providing a unified framework of data and information levels integration. On the other hand, this thesis also aims at the improvement in the development of the ontologies.

Firstly, a detailed state of the art encompassing the different production process systems, knowledge management base on ontologies, as well as decision support systems is carried out. Based on this review, the specific thesis objectives are posed. Next, a methodology is proposed for the development and use of ontologies, based on the analysis and adaptation of previously existing methodologies. Such methodology is based on the improvement cycle (PSDA), and allows for a better way to design, construct and apply domain ontologies.

The second part of this thesis is devoted to the application of the different parts of the previously proposed methodology for the development of an ontological framework in the process industry domain concerning the strategic, tactical and operational decision levels.

Next, the description of the decision areas in which the ontological framework is applied is presented. Namely, in the process control decision level, the coordination control is considered. Regarding scheduling decisions level, mathematical optimization approaches are applied. Finally, the distributed hierarchical decision level considers the mathematical optimization for decentralized supply chain networks is adopted. These decision areas and the performance of the proposed framework are studied along different case studies presented in the thesis.

On the whole, this thesis represents a step forward toward the integration among the enterprise hierarchical levels, the process and enterprise standardization and improved procedures for decision-making. The aforementioned achievements are boosted by the application of semantic models, which are currently increasingly used.

Resumen

En la actualidad, factores como la globalización del comercio, la incertidumbre del mercado y la feroz competencia implican la disminución de los márgenes de error en las empresas. Dos aspectos claves para lograrlo son la viabilidad y la competitividad de las enterprisesm, que dependen en gran medida la eficacia para la toma de sus decisiones relacionadas con sus características de fabricación, tales como eficiencia económica, la calidad del producto, la flexibilidad y fiabilidad. Por esta razón, las empresas han considerado la tarea de desarrollar mejores sistemas de gestión de la información con el fin de ayudar a los tomadores de decisiones de explotación de datos y modelos, para enriquecre y mejorar la toma de decisiones.

En este sentido, los sistemas de soporte a las decisiones deben ser mejorados con el fin de hacer frente a la gran cantidad de datos disponibles y la heterogeneidad de los métodos de modelización existentes disponibles a lo largo de los niveles jerárquicos en la estructura de la empresa. Por lo tanto, esta tesis se propone la aplicación de ontologías como herramienta de apoyo a la decisión. Las ontologias son cada vez más vistas como una tecnología clave semántica para hacer frente a los problemas creados por la heterogeneidad de los datos permitiendo asi la extracción de datos mediante el procesado del conocimiento.

El objetivo de esta tesis es contribuir al desarrollo de herramientas de apoyo para la industria de procesos. Como una herramienta de apoyo a la decisión, debe ser capaz de convertirse en un modelo sólido que interactúa entre los diferentes niveles de decisión jerárquica, proporcionando un marco unificado de datos e integración de los niveles de información. Por otra parte, esta tesis también tiene como objetivo la mejora en el desarrollo del área de ingeniería ontológica.

En primer lugar, se ha llevado a cabo un detallado estado del arte sobre los diferentes sistemas de procesos de producción, la base de la gestión del conocimiento en ontologías, así como los sistemas de soporte de decisiones. Basado en esa revision, los objetivos específicos de la tesis se plantean. A continuación, se propone una metodología para el desarrollo y uso de ontologías, con base en el análisis y adaptación de las metodologías ya existentes. Dicha metodología se basa en el ciclo de mejora (PSDA), lo que permite una mejor manera de diseñar, construir y aplicar las ontologías de dominio.

La segunda parte de esta tesis se dedica a la aplicación de las diferentes partes de la metodología propuesta anteriormente para el desarrollo de un marco ontológico en el ámbito de la industria de procesos relativos a los niveles de decisiones estratégicas, tácticas y operativas.

A continuación, la descripción de las áreas de decisión en la que se aplica el marco ontológico se presenta. Es decir, en el nivel de decision de proceso de control, el control de la coordinación se considera. En cuanto al nivel de decisiones de programación de la producción, los métodos matemáticos de optimización se aplican. Finalmente, el nivel jerárquico distribuido decisión considera la optimización matemática de las redes descentralizadas de la cadena de suministro que se adopte. Estas áreas de decisión y la desempeño del marco propuesto se estudian a lo largo de los diferentes casos de estudio presentados en la tesis.

En general, esta tesis supone un paso hacia adelante en la integración entre los niveles jerárquicos de la empresa, el proceso y la estandarización de la empresa y mejorar los procedimientos de toma de decisiones. Los logros mencionados se potencian mediante la aplicación de modelos semánticos, que actualmente se utilizan cada vez más.

Acknowledgments

I am deeply grateful to God and life for placing in my path to all those special people.

Starting with my family. To my father Máximo, example of responsibility and to my mother Guadalupe,example of dedication and love, and who have prepared me to love life. To my sisters, Mirna whose footsteps I try to follow and is an inspiration to me; Idané, who has been from birth motivation and love. To Claudia, thanks for sharing the task of parenting and mutual support. And finally I want to thank to my two big hearts Ariadna and Ian, who have enlightened and energized why to be and why to continue struggling.

Thank also the teachers whom I have worked these last 5 years, starting with Professor Alberto Lassere, who help and encouragement to continue this PhD. And most especially, to professors Luis Puigjaner and Antonio Espuña, who welcomed and guided me along this new stage of learning and development work.

Besides to my friends during this PhD phase, whom I consider as my new family, Jose Miguel, Elisabet K, Aaron, Marta, Mar, Georgios, Dejan, Marianna, Martina, Diego, Rodolfo, Victor, Miguel, Javier, Ahmed, Carlos Tostado and all those who in this I remember now, but know they are part of this cycle.

From my heart thank you.

Agradecimientos

Agradezco profundamente a todas las personas que Dios y la vida han puesto en mi camino.

Comenzando por mi familia, a mi padre Máximo mi ejemplo de responsabilidad y a mi madre Guadalupe mi ejemplo de entrega y amor, y quienes me han preparado para amar la vida, a mis hermanas, Mirna que trato de seguir tus pasos y eres una inspiración para mí, a Idané quien ha sido desde su nacimiento motivación y cariño. Claudia gracias por compartir la tarea de ser padres y el apoyo mutuo. Por ultimo a mis dos grandes corazones Ariadna e Ian, quienes han iluminado y energizado por qué estar y porque seguir esforzándome.

Agradecer también a los profesores con los que he trabajado estos últimos 5 años, comenzando por el profesor Alberto Lassere, quien ayudo e impulso a continuar el doctorado. Y de manera muy especial, a los profesores Luis Puigjaner y Antonio Espuña, quienes me acogieron y guiaron a lo largo de esta nueva etapa de aprendizaje y desenvolvimiento laboral.

Y a mis amigos de doctorado, a los cuales considero como mi familia, Jose Miguel, Elisabet K, Aaron, Marta, Mar, Georgios, Dejan, Mariana, Martina, Diego, Rodolfo, Victor, Miguel, Javier, Ahmed, Carlos Tostado y a todos aquellos que en este momento no recuerdo, pero saben que son parte de este ciclo.

De corazón gracias.

Contents

Part I Overview

Part II Ontology

Part III Framework application

Part IV Conclusions and Outlook

Appendixes

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Chemical industry overview

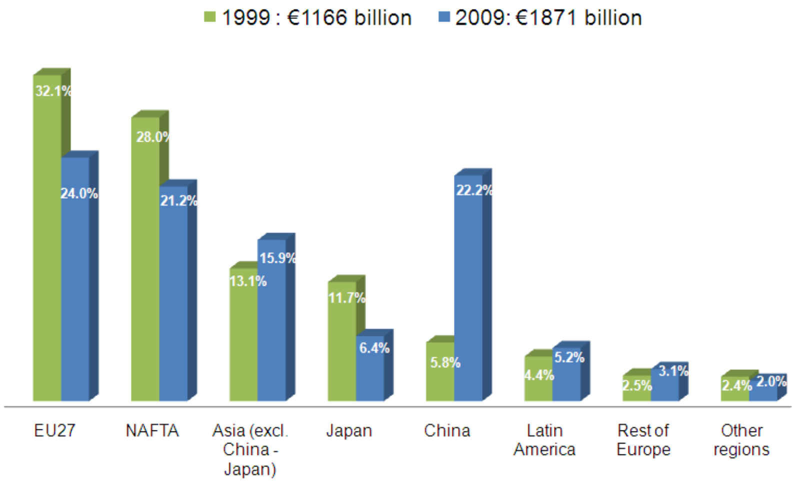
Chemistry is essential to our everyday lives. It creates opportunities for innovation, and provides a wide range of benefits for society. There has always been a strong connection between global development and innovation in chemistry: synthetic dyes were central to the development of textiles during the Industrial Revolution, and led to the birth of the pharmaceuticals industry; petrochemicals initiated the post-war plastics and materials revolution; fine and specialty chemicals offered and continue to offer a multitude of products, both for consumer items and industrial applications or processes, active ingredients for crop protection, and intermediates for pharmaceuticals.

In the 21st century, industry faces new challenges, many of which are concerns shared by society as a whole: the impact of climate change, aging populations, availability of safe drinking water, food scarcity and cost, and the security of energy supplies. With these challenges come tremendous business opportunities for the chemical industry which is uniquely placed to help to develop solutions through the creation of products that improve the quality of life, health, productivity, convenience and safety.

1.1.1 Positive social impact

The chemical industry has a positive social impact through its investment in human resources. It offers good opportunities in terms of job creation: as a science-based sector, the chemical industry requires high quality human capital and can offer good remuneration. Worldwide, about 7 million people work in the chemical industry, and taking into account indirect employment, more than 20 million people around the globe have a job connected to the chemical industry. As a sector, it is not only important in terms of size but also in terms of its features: significant capital investment, high knowledge content and qualified human resources.

It is also important for world economic and social development, as a science and technology, knowledge based industry that is essential to a sustainable world economy and improved health and nutrition. In 2009, world chemical sales is estimated in 1871 billion euros increasing 60% in 2009 compared with 1999 (see Figure 1.1) (CEFIC, 2009).



***Source**: Cefic Chemdata International*

Figure 1.1: World chemicals sales by region % of total 2009.

Industrialized countries account for a major part of world production, but the main growth centers of chemical sales and production are in emerging Asia. Total world sales in chemicals increased by around 10% from 2006 to 2007.

A major development beginning in the 1960s is the globalization of the business of chemistry, with investments by companies of many countries in production facilities in foreign countries, and the development of world markets. World economic growth, the reduction of trade barriers and taxes, as well as advances in telecommunications and air transportation, foster this globalization. Globalization of investments and markets has spread industry capital resources, technology, and managerial capabilities around the world and has resulted in the emergence of multinational chemical companies.

1.1.2 Long-range research initiative

Improved and validated scientific methods are essential to advance the understanding of chemicals management by industry, government and the general public. Celebrating its 10th anniversary in 2008, the ICCA Long-range Research Initiative (LRI) aims to ensure a sustainable and healthy future by enabling industry, regulators and society as a whole for:

- Better understand the potential impacts that chemicals may have on human health, wildlife and the environment.

- Support robust and informed decision making based on high quality information.

- Improve public confidence in decisions based on a scientific understanding of risk.

Supporting to establish a reliable framework for innovation, the work toward this goal should be developed in areas such as:

- Computational tools and models.
- Human biomonitoring.
- Intelligent testing strategies.
- Alternatives to animal use.
- Persistence and bioaccumulation.
- Exposure assessment.
- Toxicogenomics and its role in risk assessment.
- Innovations in risk assessment.

Progress in analytical methods enables the detection of very small quantities of chemicals in the body. At the same time, advanced technologies show some changes in gene expression are due to exposure to chemicals, raising concern among policy makers and the public. Industry strongly believes that the management of chemicals must be based on sound science.

The ICCA-LTR 2008 workshop held in Amsterdam, the Netherlands, focused on how technological advancements can be used to improve the science of risk assessment and how the growing body of data can be interpreted (Council, 2010).

1.2 Decision making in the enterprise

For many years companies have been developing management information systems to help the end users to exploit data and models, with the final objective of discussing and decision-making. Nowadays, global competition has made some of these decisions (related to certain manufacturing characteristics like economic efficiency, product quality, flexibility, reliability, etc.) essential for the viability of the enterprise (Venkatasubramanian et al., 2006).

Decision Support Systems (DSS) are information technology solutions that can be used to support complex decision-making and problem solving. DSS are defined as "aid computer systems at the management company level that combine data and sophisticated analytic models for support decision-making" (Simon and Murray, 2007). Classic DSS design is comprised of components for (i) sophisticated database management capabilities with access to internal and external data, information, and knowledge; (ii) modeling functions accessed by a model management system, (iii) simple user interface designs that enable interactive queries, reporting, and graphing functions, and (iv) optimization by mathematic algorithms and or intuition/knowledge. Much research and practical design effort has been conducted in each of these domains (Shim et al., 2002).

1.2.1 Modeling systems

In general terms, modeling is the attempt of devising an approximate representation of a system with the goal of providing predictions of the system's performance measures of interest. Such a representation is called a model. A model formalizes the relationship between various flows of information and can adopt different forms, from spreadsheets to mathematical programs, neural networks, expert systems among others. Furthermore a model is designed to capture certain behavioral aspects of the modeled system–those that are of interest to the analyst–in order to gain knowledge and insight into the system's behavior (Morris, 1967).

Modeling systems can be categorized from different perspectives. A general taxonomy distinguishes between transactional and analytical modeling approaches. Transactional systems are concerned with the acquisition, processing and communication of data over the enterprise. Analytical techniques introduce some reasoning to evaluate the problems, and are further classified into descriptive and normative models. Descriptive models can be used to analyze a system, but not to improve it, and provide a better understanding of internal and external functional relationships in the enterprise (included are forecasting models, cost relationships, resource utilization relationships, and simulation models). On the other hand, optimization or normative models are developed as decision-support systems to assist managers in the identification of efficient and improved decisions.

In general, descriptive and optimization algorithms can be broadly classified into equation-oriented or procedure-oriented approaches. Equation-oriented approaches involve rigorous mathematical programs, either deterministic or stochastic, constraint programming and graph theory. Procedure-oriented approaches comprise rule-based techniques, heuristics, and meta-heuristics such as simulated annealing (SA), genetic algorithms (GA), or tabu search, which are based on generic principles and schemes; they attempt to improve a given solution effectively, but the optimality and convergence are difficult to assess; there is no systematic procedure for obtaining good bounds on the attainable optimum values of the objective function (Pekny and Reklaitis, 1998).

The basis for solving a systems problem is the system representation in an adequate model, which captures the features relevant for the observer whose ultimate aim lies on decision making. Precisely, decision making in process industries results in a highly challenging task. In this area, process systems engineering (PSE) is a well established discipline of chemical engineering which covers a set of methods and tools to support decision-making for the creation and operation of the process supply chain constituting the discovery, design, manufacturing and distribution of chemical products and other process goods from a holistic approach. In order to deal with the problem complexity, it is necessary to decouple the system across a hierarchy of appropriately chosen levels. Figure 1.2 represents a hierarchy of particular modeling systems that can be distinguished in the broad area of PSE based on the temporal scale and the level of decision. The need to integrate the different modeling approaches in a hierarchical decision-support system makes necessary the use of consistent terminology and concepts to improve the communication and collaboration tasks over the entire system.

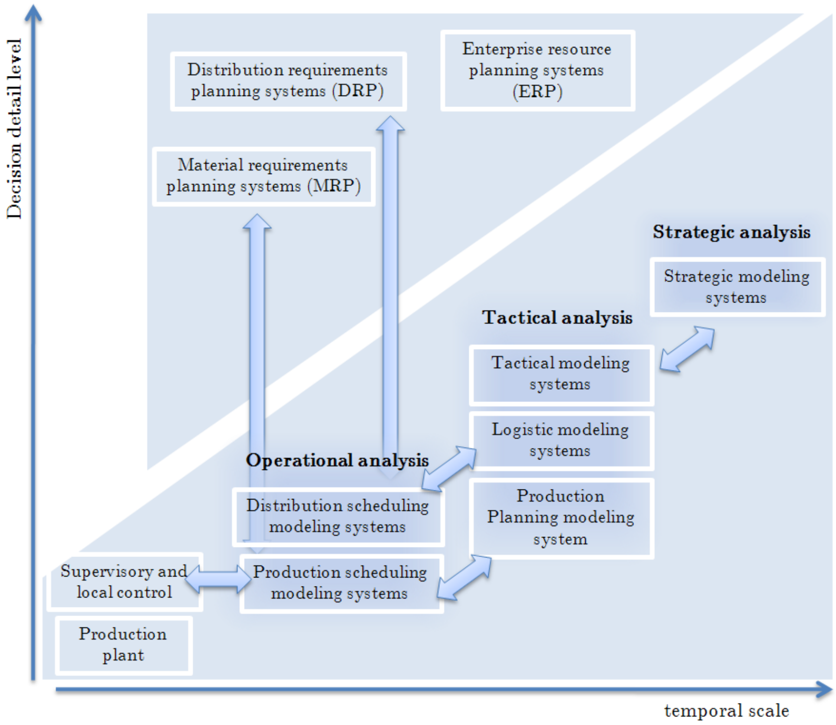

Decision detail level
Distribution requirements planning systems (DRP)
Enterprise resource planning systems (ERP)
Material requirements planning systems (MRP)
Strategic analysis
Strategic modeling systems
Tactical analysis
Tactical modeling systems
Logistic modeling systems
Operational analysis
Distribution scheduling modeling systems
Production Planning modeling system
Supervisory and local control
Production scheduling modeling systems
Production plant
temporal scale

Figure 1.2: Hierarchy of modeling systems

1.3 Knowledge management

A knowledge management of a specific domain can usually consider its knowledge representation and is organized by five principles: i) a surrogate; ii) a set of ontological commitments; iii) a fragmentary theory of intelligent reasoning; iv) a medium for efficient computation; v) a medium of human expression (Davis et al., 1993).

The competitiveness of companies depends heavily on how they exploit their corporate knowledge and memory. One of the enterprise basic commitments to ensure a sustainable success is to manage its knowledge. It refers to the development and exploitation of the organization's tangible and intangible knowledge resources. Organizational knowledge management is concerned with realizing the value of this "intellectual capital", which exists as: tangible assets (such as patent licenses and information held in databases on customers, suppliers, products and competitors, etc) and intangible assets (such as the skills, experience and knowledge of people within the organization) (Apostolou et al., 2008).

Besides, the support for information and knowledge exchange is a key issue in the enterprise information system. The exponential growth of on-line information on intranets and the web leads to information overload. To cut down on the time wasted in searching and browsing, and reduce associated user frustration, much more selective user access is needed.

Most information in modern electronic media is mixed-media and rather weakly structured. Finding and maintaining information is a hard problem in weakly structured representation media. Increasingly, companies realize that their intra-nets are valuable repositories of corporate knowledge. But with the now rapidly increasing volumes of information, turning this into useful knowledge has become a major problem. Knowledge Management is about leveraging corporate knowledge for greater productivity, value, and competitiveness (Schreiber et al., 2000).

Due to Internet-enhanced globalization, many organizations are increasingly geographically dispersed and organized around virtual teams. Such organizations need knowledge management and organizational memory tools that encourage users and foster collaboration while capturing, representing and interpreting corporate knowledge resources and their meaning.

Knowledge management tools can be based on several technologies such as distributed data bases, ontologies, networks maps. In section 1.3.1 a general overview of ontologies is given as a developed tool in this thesis.

1.3.1 Ontology definition

Ontologies are increasingly seen as a key semantic technology for addressing heterogeneities and mitigating the problems they create and for enabling semantics-driven knowledge processing. Ontologies are formal structures enabling acquiring, maintaining, accessing, sharing and reusing information (Gruber, 1993; Fensel, 2003). Knowledge management systems benefit from ontologies that semantically enrich information and precisely define the meaning of various information artifacts.

From the essence of enterprise modeling, we could say that it plays a critical role in this integration, enabling better analysis of their performance, and management of their operations. An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise. The role of an enterprise model is to achieve model-driven enterprise design, analysis, and operation (Fox et al. (1997); Fox and Gruninger (1998)).

Ontology is an important emerging discipline that has a significant potential to improve information organization, management and understanding. It has a crucial role to play in enabling content-based access, interoperability, communications, and providing qualitatively new levels of services on the next wave of Semantic Web and other research domains.

Ontologies have been developed in order to facilitate knowledge sharing and reuse. They are a popular topic in various research communities, such as knowledge engineering, natural language processing, cooperative information systems, information integration, software agents, and knowledge management. Generally speaking, Ontologies provide:

- A shared and common understanding of a domain; this domain can be communicated among people and across application systems.

- An explicit conceptualization that describes the semantics of the data.

1.4 Research scope

The main objective of this thesis is to bridge systems engineering processes domain (chemical process) with computer science domain (artificial intelligence, ontological engineering and semantic web). This work should offer the required effective techniques for rapid integration of communication and management of multiple systems. For this reason, the proposed system should encompass standards (process standards, informatics language standards, and ontological standards), which should give the strength for its future validation, sharing and application.

Systems engineering processes are comprised by different independent activities (such as pre-formulation and new process development, supply chain management, scheduling, process control, fault analysis, etc.). These activities are included within a functional hierarchical model (Purdue reference model (Williams, 1989)) which includes: business planning & logistics, manufacturing operations & control, and batch, continuous, or semi-continuous control.

On the other hand with regard to computer science domain, the model should define the hierarchical levels at which decisions are made, and such model should become the basis of a framework for developing a multi-scale decision support model. Thus, this model would be supported by the ontology as a data, information and knowledge system management.

1.5 Thesis outline

This thesis has been structured following a logical sequence stemming from the general concepts to the practical application as shown in Figure 4.7, and explained next.

Firstly, in Part I, the state of the art about the different production process systems, knowledge management based on ontologies, as well as decision support systems are presented in Chapter 2. Thus, at the end of this Chapter the specific thesis objectives are posed. Next, Chapter3 sets the methodology proposed for the development and use of ontologies, based on the analysis and adaptation of previously existing methodologies. Such methodology is based on the improvement cycle (PSDA), allowing a better way to design, construct and apply domain ontologies.

Part II of this thesis is devoted to the application of the different parts of the previously proposed methodology for the development of an ontological framework in the process industry domain concerning the strategic, tactical and operational decision levels.

Thus, Part III presents the application of the ontological framework at decision levels of the enterprise. Specifically, Chapter 5 describes the decision making tools adopted in this thesis at each level of the hierarchical structure. A the process control decision level, the coordination control is considered. Regarding scheduling decisions, mathematical optimization approaches are applied. Finally, the distributed hierarchical decisions considering mathematical optimization for decentralized supply chain networks is adopted. These decision areas and the performance of the proposed framework interaction are studied along the different case studies presented in Chapter 6.

Finally, Part IV summarizes the conclusions derived from this research, and proposes the future work that may be derived.

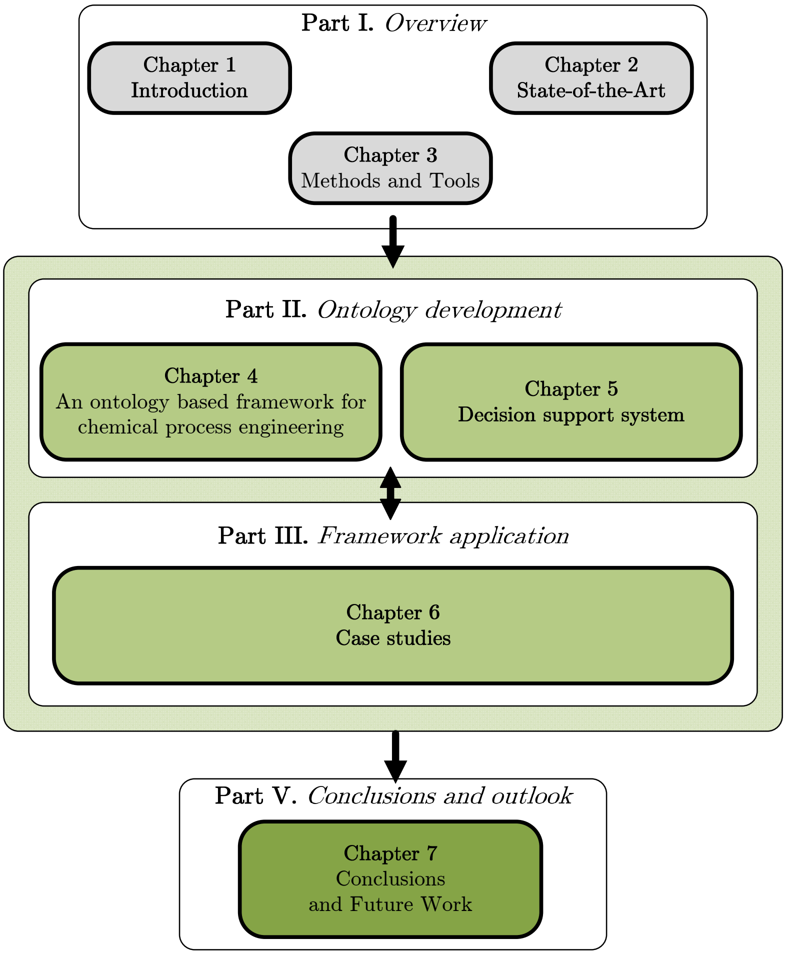


Figure 1.3: Thesis outline.

Chapter 2

State of the Art

In this chapter, the description of the current state of the art regarding production process systems, knowledge management tools based on ontologies, and decision support systems, is detailed. Thus the thesis objectives based on the challenges identified in the literature are posed.

2.1 Production process systems

In this section, the major production process patterns found in chemical process industries are presented. Production processes can be classified as continuous, semicontinuous and batch. This thesis' work focuses primarily on batch production processes because of their greater complexity (see section 2.1.3 for details). However, one of the most important goals is flexibility of adaptation to any other kind of process. The physical model differences of production process types are shown in the Figure 2.1.

2.1.1 Continuous processes

In a continuous task, materials and products are produced continuously along the time period of the task, and processing rate can be either fixed or within a certain range. The continuous production process is adopted by most oil and gas industries and petrochemical plants and in other industries, such as the float glass industry, where glass of different thickness is processed in a continuous manner. Manufacturing firms often run continuous processes to perform some production steps yielding intermediates and/or final products.

Many fast moving consumers good manufacturing companies produce a moderate number of intermediates that are combined in many different ways to generate an enormous variety of end products. To do that, such companies usually run continuous production plants in a make-to-stock environment. The process structure, includes

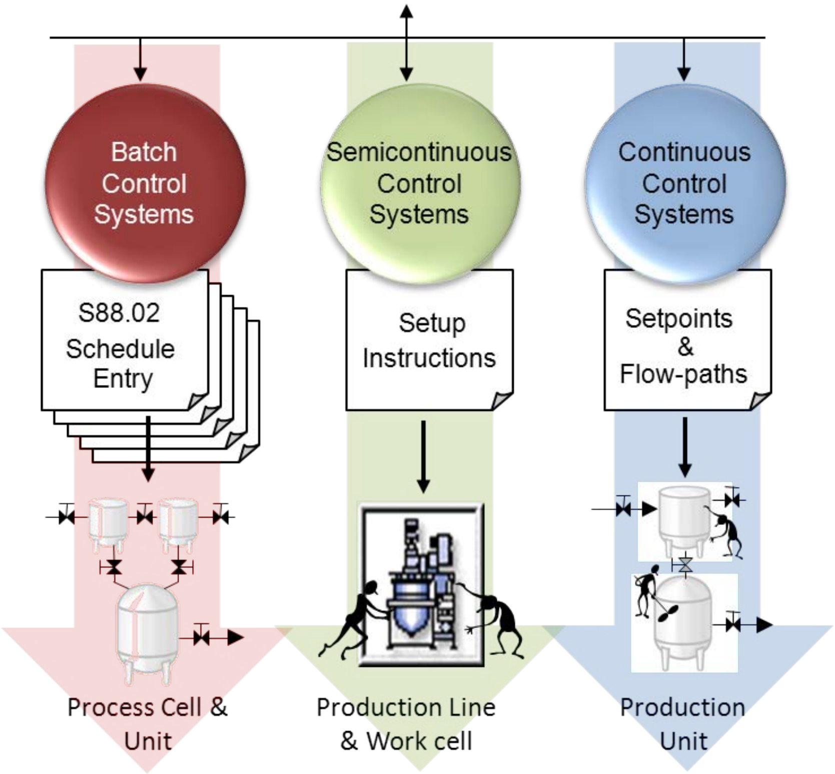


Figure 2.1: Production process systems.

a fabrication area yielding basic intermediates that are stocked in a large middle storage space, and a packing sector where finished products, usually comprising several intermediates, are manufactured.

2.1.2 Semi-continuous processes

In a semi-continuous task, either the raw materials or the products are fed continuously; whereas the other is loaded/discharged at a time. Semi-continuous processing offers a more customized operation for highly dynamic and uncertain environments. Semi-continuous operations are characterized by their processing rate, running continuously with periodic start-ups and shutdowns for frequent product transition. The processing times of semi-continuous processes are relatively long periods of time called campaigns, each dedicated to the production of a single product. Typical campaign lengths range from a few hours to several days. Most process plants in the chemical industry combine continuous operations and batch processes in their product processing routes thus working in semi-continuous mode, since production becomes more flexible and equipment can be more efficiently utilized.

2.1.3 Batch processes

In a batch task, materials are fed at the start of the task; and after a certain time, products are produced at once at the end. In a production batch plant environment, the presence of unpredictable events not only related to external market factors but also to the intrinsic plant operation itself, such as equipment breakdowns and variable operation times, is usually unavoidable. Even more, due to the existent lack of integration between the different control levels (control levels of the Purdue Reference Model) (Williams, 1989) the decision support systems task becomes harder.

A crucial opportunity for batch process improvement and optimization is to develop information structures which (i) streamline data gathering, and even more, (ii) are capable of integrating transactional a system with the analytical tools developed. Current trends in electronics, computer science & artificial intelligence, and control system technology are providing the technical capability to greatly facilitating the development of a support system for multi-level decision-making. This information structure must become increasingly agile and integrated across the batch process functions.

M ode of operation The operation of most chemical plants is influenced by how the orders are organized. If orders are relatively long-term, plants can operate in campaign mode, with all the resources of the plant dedicated to a subset of products for a period of time (campaigns). Thus, the control of the plant's production is simplified and cyclic production can be established for a single product (or multi product) campaign, in which many batches of identical sequences of the same product (or products) are produced.

Conversely, if demand is not reliable, the production plant is based primarily on manufacturing the available orders. This forces planning on a rather short-time horizon, and a regular pattern of production cannot be established. The situation gives rise to the problem of scheduling operations in the short-term.

2.2 Standards for integration

2.2.1 Process Standards

Process standards are guided by the ANSI/ISA standards playing a role in the work of instrumentation and automation professionals, besides they are recognized by the American National Standards Institute (ANSI). ANSI/ISA standards cover a wide range of concepts of importance to instrumentation and automation at manufacturing. ANSI/ISA has standards committees for symbols and nomenclature used within the industry, safety standards for equipment in non-hazardous and hazardous environments, communications standards to permit interoperability equipment availability from several manufacturers, and additional committees for standards on many more technical issues of importance to the industry.

The ANSI/ISA-88 (International Society for Measurement and Control, 1995; Measurement and Control, 2001; International Society for Measurement and Control, 2003, 2006, 2007a) defines standards and recommended practices for the design and specification of batch control systems as used in the process control industries. Unlike other, the ANSI/ISA-88 standards are not a compliance standard, and they are defined

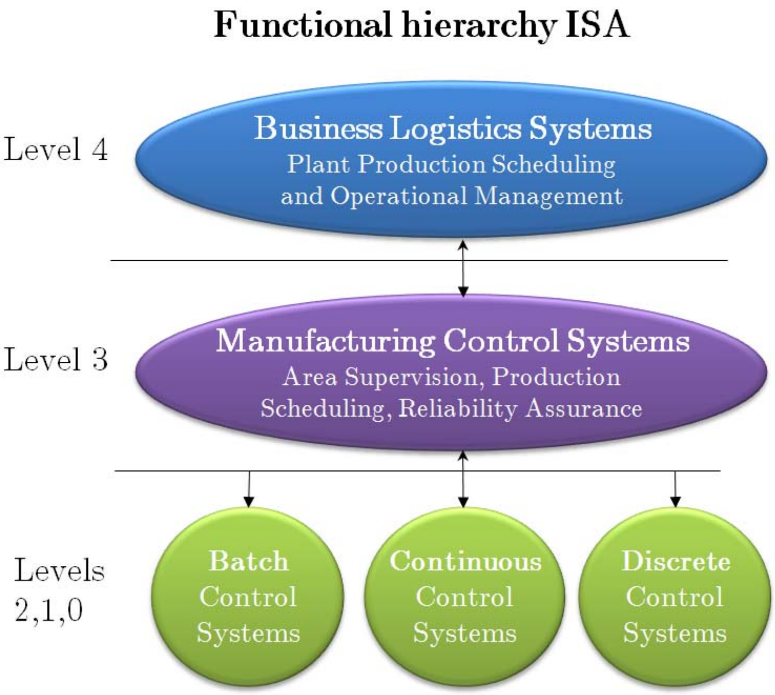

Functional hierarchy ISA
Level 4
Business Logistics Systems
Plant Production Scheduling and Operational Management
Level 3
Manufacturing Control Systems
Area Supervision, Production Scheduling, Reliability Assurance
Levels 2,1,0
Batch Control Systems
Continuous Control Systems
Discrete Control Systems

Figure 2.2: Hierarchical model control & business systems.

as a guideline which contains the preferred term for systems and software based on batch process requirements.

The application of ANSI/ISA-88 to the development of a batch control system can facilitate the implementation of the project, as it gives engineers a clear set of terms to describe flow sheets and control schemes and can considerably reduce the time required to implement the system. it has been very successful in its fundamental task of explaining what batch control is all about, and in cutting the time needed to develop and configure software (Brandl and Emerson, 2003).

ANSI/ISA-95 (International Society for Measurement and Control, 1999) is a definition of the functions associated with the interface between control functions and enterprise functions. Even more it functions as a definition of the information which is shared between control functions and enterprise functions.

When the ANSI/ISA-88 series and the ANSI/ISA-95 series of standards are used within the same plant-wide automation system in an enterprise, it is necessary to align the various definitions in both standards in order for industry stakeholders to reap the intended benefits. In table 2.1 a comparison of approaches and coverage between the ANSI/ISA-88 and ANSI/ISA-95 is shown.

When the ANSI/ISA-88 and ANSI/ISA-95 standards are applied together in an application or project, some of the terminologies, models and key definitions in these two sets of specifications need to be aligned to assist the users. A harmonization task group with members from the ANSI/ISA-88 and ANSI/ISA-95 committees has generated this technical report to document and map the overlaps, gaps, similarities and

Table 2.1: Comparison of approaches and coverage.

Item	ISA-95	ISA-88
Orientation	Definition of work flow and information exchange for Manufacturing Operations Management.	Physical work execution for Batch and other types of manufacturing.
Conceptual Basis relative to Manufacturing Management Functions	Flexible structure of manufacturing management functions that interacts with business requirements.	Acknowledges but does not directly address manufacturing management functions.
Conceptual Basis relative to Process Control	Stops short of directly addressing most traditional process control activities.	Well-defined equipment-oriented process control structure and function hierarchies extending to the bits and pieces of the manufacturing equipment itself.
Primary areas of concern	In the way most people describe a manufacturing enterprise, addresses business functionalities and applications at a level below enterprise business systems but above the physical manufacturing equipment.	Addresses a lower level, directing, controlling and coordinating the people and equipment that carry out the physical transformation of raw materials into final or intermediate products.
Affected Industries	Spans all types of manufacturing.	Written primarily in terms of batch manufacturing, but is often applied in other types of manufacturing.

differences in these concepts, terms and definitions (International Society for Measurement and Control, 2007b). Figure 2.3 shows the interaction between ANSI/ISA-95 for enterprise-to-control system integration and ANSI/ISA-88 for control system integration in the enterprise control system.

2.2.2 Enterprise integration

To remain competitive, enterprises must become increasingly agile and integrated across their functions. Enterprise models play a critical role in this integration, enabling better designs for enterprises, analysis of their performance, and management of their operations.

An enterprise model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise. It can be both descriptive and definitional spanning what is and what should be. The role of an enterprise model is to achieve model-driven

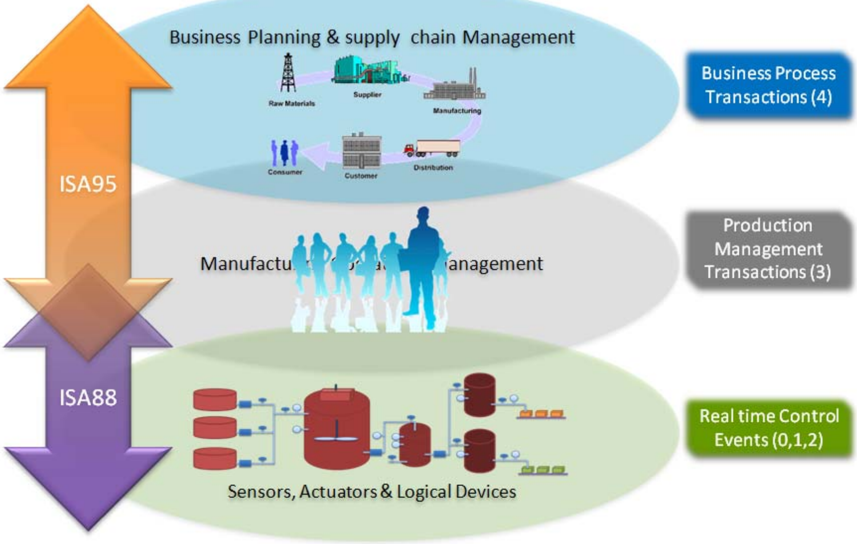

Business Planning & supply chain Management
Raw Materials
Supplier
Manufacturing
Distribution
Customer
Consumer
ISA95
Manufactu anagement
ISA88
Sensors, Actuators & Logical Devices
Business Process Transactions (4)
Production Management Transactions (3)
Real time Control Events (0,1,2)

Figure 2.3: ANSI/ISA interaction at the enterprise control system.

enterprise design, analysis, and operation.

From a design perspective, an enterprise model should provide the language used to explicitly define an enterprise. From an analysis perspective we need to be able to explore alternative models in the enterprises spanning organization structure and behavior. From an operations perspective, the enterprise model must be able to represent what is planned, what might happen, and what has happened. It must supply the information and knowledge necessary to support the operations of the enterprise to provide answers to questions commonly asked in the performance of tasks.

Achieving integration requires an information infrastructure that supports the communication of information and knowledge, the making of decisions, and the coordination of actions. At the heart of this infrastructure lies a model of the enterprise.

The problem faced along last years was that the legacy systems that support enterprise functions were created independently and, consequently do not share the same enterprise models. It is called this the correspondence problem. Although each enterprise model might represent the same concept, for example, activity, they will have a different name, for example, operation versus task. Consequently, communication among functions is not possible without at least some translation, but no matter how rational the idea of renaming them is, organizational barriers impede it. Further, these representations lack an adequate specification of what the objects (terminology) mean (that is, semantics). Instead, concepts are poorly defined, and their interpretations overlap, leading to inconsistent interpretations and uses of the knowledge. Finally, the cost of designing, building, and maintaining a model of the enterprise is large. Each tends to be unique to the enterprise; objects are enterprise specific.

The industrial systems in order to be able to produce that plant's products at maximum overall profit for the company involved with at minimum cost have endeavored to integrate the operating units of the plant.

Some work and efforts to accomplish this have been developed, and can be divided by trends.

- The first trend was based on techniques that intended to achieve closely coupled production units, minimize in-process inventories and work in progress, and make maximum use of in-plant energy sources to supply plant energy needs.
- A second trend was based on techniques that promote the use of automatic control in its broadest sense (including dynamic control, scheduling and the closure of information loops) to integrate all aspects of the plant's operations including closing the information loops within the plant.
- Finally, current trends are based in electronics, computer science and control systems that include: distributed, digital, microcomputer based, the dynamic control systems; standard real-time programming languages; standardized high speed serial data links; and corresponding major developments in database management techniques. This last trend has only been possible thanks to the advent of the modern computer that can handle the enormous computational involved in carrying out functions in real time.

Most of these techniques and trends, latter will result in partial large scale, hierarchically arranged computer systems integrating the plant management, plant production scheduling, inventory management, individual process optimization, and unit process control for all of the plant's operating units treated as a whole. In addition, some features such as the lack of unit coordination, the lack of dynamic response, and the lack of market sensitivity may also be covered by the application of the aforementioned techniques.

The Purdue reference model provides an "environment" for discrete parts manufacturing and forms the basis for the other models. Certain activities have been identified which are directly related to shop floor production. A six level hierarchical model was selected to represent those activities (see Figure 2.4). It is quite likely that specific applications may require more or fewer than six levels. But, six was deemed sufficient for the purposes of identifying where integration standards are required. The following list shows the name of each level and gives its major responsibility.

- Level 6 Enterprise - Corporate Management (External Influences)
- Level 5 Facility - Planning Production
- Level 4 Section-Material/Resource Supervision
- Level 3 Cell - Coordinate Multiple Machines
- Level 2 Station- Command Machine Sequences
- Level 1 Equipment-Activate Sequences Of Motion (Plant Machinery And Equipment)

These activities apply to manual operations, automated operations, or a mixture of the two at any level. The Purdue scheduling and control perception, do not include levels 1 and 6. It is important to mention that the 6 tasks are easily subdivided into those related to production scheduling, control enforcement, systems coordination and reporting, and reliability assurance. In the context of any large industrial plant, or of a complete industrial company based on one location, the tasks would be carried out at each level of the hierarchy.

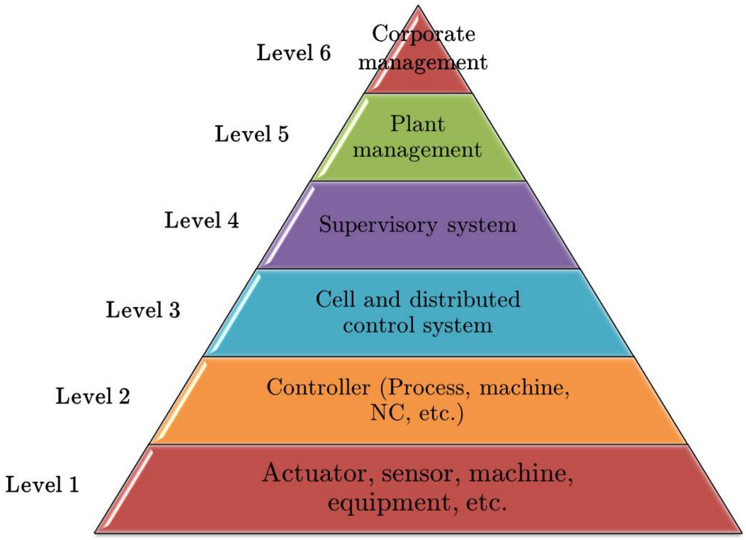

Level 6
Corporate management
Level 5
Plant management
Level 4
Supervisory system
Level 3
Cell and distributed control system
Level 2
Controller (Process, machine, NC, etc.)
Level 1
Actuator, sensor, machine, equipment, etc.

Figure 2.4: Hierarchical Purdue reference model.

In this way, the international Purdue workshop (Williams, 1989) on industrial computer systems, carried out such a development for computer integrated manufacturing (CIM) as applied to all industries. The CIM reference model was an enormous step toward successfully system integration. A reference model is a previously agreed-upon or "standard" definitive document or conceptual representation of a system. The reference model defines requirements common to all implementations but is independent of the specified requirements of any particular implementation.

The CIM Reference Model is thus a reference for computer integrated manufacturing. It is a detailed collection of the generic information management and automatic control tasks and their necessary functional requirements for the manufacturing plant.

Nevertheless the CIM reference model committee has been limited to the elements of the integrated information management and automation system. This means that the company's management (future planning function); financial; purchasing; research, development and engineering; and marketing and sales have all been treated as external influences.

2.2.3 Information and knowledge management standards

Information and knowledge management standards, are in this work guided by The World Wide Web Consortium (W3C) which is an international consortium to develop informatics Web standards. W3C primarily pursues its mission through the creation of Web standards and guidelines designed to ensure long-term growth for the Web as the actuality and future of the informatics framework.

One of the most important contributions is the Extensible Markup Language (XML) which describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) since 1996. It was chaired by Bosak (1997) with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

This specification, together with associated standards (Unicode and ISO/IEC 10646 for characters, Internet BCP 47 and the Language Subtag Registry for language identification tags), provides all the information necessary to understand XML Version 1.0 and construct computer programs to process it.

2.3 Knowledge management: ontologies

Due to Internet-enhanced globalization, many organizations are increasingly geographically dispersed and organized around virtual teams. Such organizations need knowledge management and organizational memory tools that encourage users and foster collaboration while capturing, representing and interpreting corporate knowledge resources and their meaning.

How to manage this knowledge has become an important issue in the past few decades, and the knowledge management (KM) community has developed a wide range of technologies and applications for both academic research and practical applications. In addition, KM has attracted much effort to explore its nature, concepts, frameworks, architectures, methodologies, tools, functions, real world implementations in terms of demonstrating KM technologies and their applications. Therefore, the new era of information and communication technology plays important roles not only in electronic commerce but also in knowledge management. Due to this technological advance, KM expands in a rapidly manner. In this sense, there is some classification of KM technologies highlighting seven categories, according to Liao (2003): KM framework, knowledge-based systems (KBS), data mining (DM), information and communication technology (ICT), artificial intelligence (AI)/expert systems (ES), database technology (DT), and modeling.

Specifically, information and communication technology (ICT) has been considered as a technology used toward decision support. In todays information economy, rapid access to knowledge is critical to the success of many organizations. An information and communication technology (ICT) infrastructure provides a broad platform for exchanging data, coordinating activities, sharing information, emerging private and public sectors, and supporting globalization commerce, all based on powerful computing and network technology. Information computing offers powerful information processing abilities, and the network provides standards and connectivity for digital integration. The ICT encompasses ontologies.

Ontology is the knowledge integration of different representations of the same piece of knowledge at different levels of formalization. The experts who participate in the ontology process are allowed to use their own terminology, facilitating knowledge integrations with cooperative tools (Fernandez-Breis and Martinez-Bejar, 2000). Ontologies are increasingly seen as a key semantic technology for addressing heterogeneities and mitigating the problems they create and for enabling semantics-driven knowledge processing. Thus, ontologies enrich information and precisely define the meaning of various information artifacts. As a whole, ontologies are formal structures enabling acquiring, maintaining, accessing, sharing and reusing information (Gruber, 1993; Fensel, 2003).

2.3.1 Ontology design principles

Here there is a summary of some design criteria and a set of principles that have been proved useful for the development of ontologies.

- Clarity and bbjectivity (Gruber, 1993), which means that the ontology should provide the meaning of defined terms by providing objective definitions and also natural language documentation.
- Completeness (Gruber, 1993), which means that a definition expressed by a necessary and sufficient conditions is preferred over a partial definition (defined only by a necessary or sufficient condition).
- Coherence (Gruber, 1993), to permit inferences that are consistent with the definitions.
- Maximize monotonic extendibility (Gruber, 1993), It means that new general or specialized terms should be included in the ontology in such a way that the revision of existing definitions is not required.

- Minimal ontological commitments (Gruber, 1993), which means making as few claims as possible about the world being modeled, which means that the ontology should specify as little as possible to the ontology freedom to specialize about the meaning of its terms.
- Ontological distinction principle (Borgo et al., 1996) which means that classes in an ontology should be disjoint. The criterion used to isolate the core of properties considered to be invariant for an instance of a class is called the Identity Criterion.
- Diversification of hierarchies to increase the power provided by multiple inheritance mechanisms (Vega et al., 1996). If enough knowledge is represented in the ontology and as many different classification criteria as possible are used, it is easier to enter new concepts (since they can be easily specified from the pre-existing concepts and classifications criteria) and to inherit properties from different points of view.
- Modularity (Bernaras et al., 1996) to minimize coupling between modules.
- Minimize the semantic distance between sibling concepts (Vega et al., 1996). Similar concepts are grouped and represented as subclasses of one class and should be defined using the same primitives, whereas concepts which are less similar are represented farther apart in the hierarchy.
- Standardize names whenever is possible (Vega et al., 1996).

Finally, the issue of ease of reuse is the focal point of study in many research projects. In artificial intelligence and informatics science, ontologies were born to help in knowledge reuse and sharing: reuse means building new applications by assembling components already built, while sharing occurs when different applications use the same resources. Both reuse and sharing have the advantage of being cost, time and resources effective.

2.3.2 Ontology programming conventions

Next, the basic terms and constructs are introduced and explained in order to describe the main programming characteristics of an ontology.

- A taxonomy is a set of concepts, which are arranged hierarchically. A taxonomy does not define attributes of these concepts. It usually defines only the "is-a" relationship between the concepts. In addition to the basic is-a relation, the part-of relation may also be used.
- A vocabulary is a language dependent set of words with explanations/ documentation. It seeks universality and formality in a local context.
- Axioms are the elements which permit the detailed modeling of the domain and constitute the "core knowledge" that you must assume to be true during reasoning. Axioms is often used to denote coherent statements that can be made in RDFS/OWL.

- A class or type is a set of objects. Each of the objects in a class is said to be an instance of the class. In some frameworks an object can be an instance of multiple classes. The top classes employed by a well developed ontology derive from the root class object, or thing, and they themselves are objects, or things. Each of them corresponds to the traditional concept of being or entity.
- Every class and every individual has a unique identifier, or name. The name may be a string or an integer and is not intended to be human readable.
- By conceptualization we mean a set of concepts, relations, objects and constraints that define the domain in question.
- An object-oriented database schema defines a hierarchy of classes and attributes and relationships of those classes.
- Objects that are not classes are referred to as individuals. Thus, the domain of discourse consists of individuals and classes, which are generically referred to as objects. Individuals are objects which cannot be divided without losing their structural and functional characteristics. They are grouped into classes and have slots.
- Objects have associated a set of own slots and each own slot of an object has associated with it a set of objects called slot values. Slots can hold many different kinds of values and can hold many at the same time. They are used to store information, such as name and description, which uniquely define a class or an individual.
- Relations operate among the various objects populating an ontology. In fact, it could be said that the conjunctions of any articulated ontology is provided by the network of dependency relations among its objects. The class-membership relation that holds between an instance and a class is a binary relation that maps objects to classes. The type-of relation is defined as the inverse of instance-of relation.
- Inheritance through the class hierarchy means that the value of a slot for an individual or class can be inherited from its super class.
- A knowledge base is a collection of classes, individuals, slots, slot values, facets, and facet values. A knowledge base is also known as a module.

2.3.3 Ontology methodologies

It is seen for many people in the ontological field that the ontology building process is a craft rather than an engineering activity. The construction of an ontology is a time-consuming and complex task.

Nowadays, numerous ontologies are being developed and used in various research areas. Each development project usually follows its own set of principles, design criteria and phases in the ontology development process. The absence of standards guidelines and methods hinders the development of shared and concentrated ontologies within and between projects, the extension of a given ontology by others and its reuse in other ontologies and final applications.

There are also ontology servers that collect a number of ontologies. Even if it is widely recognized that constructing ontologies, or domain models, is an important step in the development of knowledge-based systems (KBS), what is lacking is a consensus for a uniform approach in designing and maintaining these ontologies.

Various methodologies exist to guide the theoretical approach taken, and numerous ontology building tools are available. The problem is that these procedures have not coalesced into popular development styles or protocols, and the tools have not yet matured to the degree one expects in other software instances.

However, it exist a small but growing number of methodologies that specifically address the issue of the development of ontologies. Some of those are the following:

- TOVE and enterprise modeling:

  The Toronto Virtual Enterprise (TOVE) is a deductive enterprise model (EM), an extension of a generic enterprise model. An enterprise Model is a computational representation of the structure, activities, processes, information, resources, people, behavior, goals and constraints of a business, government or other enterprise. It can be both descriptive and definitional. The role of an enterprise model is to achieve model-driven enterprise design, analysis and operation. The TOVE group developed a methodological approach for the construction of an EM based on the definition of (Gruninger and Fox, 1995; Uschold and Gruninger, 1996): motivating scenarios, informal competency questions, terminology specification, formal competency questions, axiom specification, completeness theorems. In common with most recent Knowledge Base Systems development methodologies, the Enterprise approach distinguishes between informal and formal phases of ontology construction. In fact, the Enterprise ontology does not explicitly deploy a formal evaluation procedure; this was the main focus of the methodology used in the context of the TOVE project.

  Given the basic work on construction and evaluation methodologies by Uschold and Gruninger (1996), others have focused on the preliminary phases of construction.

- Methontology:

  Methontology, on the other hand, provides support for the entire life-cycle of ontology development. It enables experts and ontology makers who are unfamiliar with implementation environments to build ontologies from scratch. Initially described in Gómez-Pérez and Fernández (1996) and then updated in Fernandez-Lopez et al. (1997), Methontology identify the following activities in the development of an ontology: specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, documentation. The life cycle of the ontology is based on the refinement of a prototype and ends with a maintenance state. The most distinctive aspect of Methontology is the focus on its maintenance. The environment for building ontologies using the Methontology framework is called ODE (Ontology Design Environment). ODE is a software tool to specify ontologies at the knowledge level. ODE allows developers to specify their ontology by filling in tables and drawing graphs. It has a module which automatically translates the specification of the ontology into target languages.

- The SENSUS-based methodology:

It was developed at the IST (Information Sciences Institute) to provide a broad-based conceptual structure for developing machine translators. SENSUS uses a top-down approach for deriving domain specific ontologies from giant ontologies and is semi- application dependent (Knight and Luk, 1994). Application-semidependent means the possible scenarios of ontology use are identified in the specification stage.

- The On-To-Knowledge methodology:

  It includes the identification of goals that should be achieved by knowledge management tools and is based on an analysis of usage scenarios. The steps proposed by the methodology are: kickoff, where specified, competency questions are identified, potentially reusable ontologies are studied and a first draft of the ontology is built; refinement, where a mature and application-oriented ontology is produced; evaluation, where requirements and competency questions are checked, and the ontology is tested in the application environment; and ontology maintenance. This method is application dependent this indicates that the process by which the ontology is built is totally independent of the uses to which it will be put in knowledge-based systems, agents, etc. (Sure and Studer, 2002).

2.3.4 Ontology types

This section does not seek to give an exhaustive typology of ontologies, nevertheless it presents the most commonly used types of ontologies and their basic principles.

Basically, the following categories are identified: knowledge representation ontologies, meta-ontologies, domain ontologies, tasks ontologies, domain-task ontologies, application ontologies, index ontological tell and ask ontologies, etc.

- Knowledge representation ontologies (van Heijst et al., 1997) capture the representation primitives used to formalize knowledge in knowledge representation paradigms. The most representative example of this kind of ontologies is the Frame-ontology (Gruber, 1993) which captures the representation primitives (classes, instances, slots, facets, etc.) used in frame-based languages.

- General/common ontologies (Mizoguchi et al., 1995) include vocabulary related to things, events, time, space, causality, behavior, function, etc. (van Heijst et al., 1997), which are reusable across domains. The most representative example could be a teleology ontology which would include the term part-of.

- Domain ontologies (Mizoguchi et al., 1995) (van Heijst et al., 1997) are reusable in a given domain. They provide vocabularies about the concepts within a domain (i.e., scalpel, scanner in a medical domain) and their relationships, about the activities that take place in that domain (i.e., anesthetize, give birth), and about the theories and elementary principles governing that domain.

In addition, according to the degree of formality of the ontology, Uschold and Gruninger (1996) identified, in their overview of this field, the following types: highly informal, semi-informal, semi-formal, rigorously formal. In the informal classes there are more or less structured definitions in natural language. In the formal classes there are ontologies defined through artificial formal languages (e.g., Ontolingua) or first order theories with formal semantics, theorems and proofs of such properties as soundness and completeness (Gómez-Pérez, 2007).

2.3.5 Ontology languages

Different ontology languages provide different facilities. Any language used to codify ontology-underpinned knowledge should be expressive, declarative, portable, domain independent and semantically well defined. Next, the most representative ontology languages are presented:

- Ontolingua

  The original Ontolingua language, as described by Gruber (1993), was designed to support the design and specification of ontologies with a clear logical semantics. To accomplish this, Gruber (1993) started from KIF (knowledge interchange format) and extended it with additional syntax, to capture intuitive bundling of axioms into definitional forms with ontological significance, and a frame ontology to define object oriented and frame-language terms. The ontolingua Server has extended the original language in two ways. First, it provides explicit support for building ontological modules that can be assembled, extended, and refined in a new ontology. Secondly, it makes an explicit separation between ontologies presentation (the manner in which KIF axioms are viewed and manipulated by a user) and semantics (the underlying meaning).

  The original ontolingua language provided limited support for defining ontological modules in the form of a tree of named ontologies. Users found this simple model to be inadequate in several ways. Furthermore, the module system did not have a clearly defined semantics; this was in sharp conflict with the basic goals of the language.

  The separation of presentation and semantics has always been implicit in ontolingua's translation approach to sharing ontologies. In the current system, however, the explicit recognition of this distinction has become a key notion. The semantics of an ontology is always defined by a set of KIF axioms. In ontolingua, the semantics is always simple, clear, and unambiguous. The presentation, in the ontolingua Server's browsing and editing environment, is tailored for object-oriented or frame-language descriptions of the world.

  Farquhar et al. (1997) guarantee that each statement corresponds unambiguously to a KIF axiom. The vocabulary used in the presentation is defined in the Frame Ontology. The Frame Ontology defines terms including class, subclass-of, slot, slot-value-type, slot-cardinality, facet and so on. If an ontology is defined using this vocabulary, the ontolingua server can present it in a user-friendly form.

- CycL representation language

  The CycL representation language, is a large and flexible knowledge representation language (Lenat and Guha, 1989). It is essentially an extension of first-order predicate calculus, with extensions to handle equality, default reasoning, skolemization, and some second-order features. For example, quantification over predicates is allowed in some circumstances, and complete assertions can appear as intentional components of other assertions. CycL uses a form of circumscription, includes the unique names assumption, and can make use of the closed world assumption where appropriate.

- LOOM

LOOM is a language and environment for building intelligent applications. The heart of LOOM is a knowledge representation system that is used to provide deductive support for the declarative portion of the LOOM language (MacGregor, 1991). Declarative knowledge in LOOM consists of definitions, rules, facts, and default rules. A deductive engine called a classifier utilizes forward-chaining, semantic unification and object-oriented truth maintenance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing.

- Generic frame protocol

  The Generic Frame Protocol (GFP) (Karp et al., 1995), jointly developed at SRI International and Knowledge Systems Laboratory of Stanford University, provides a set of functions that support a generic interface to underlie frame representation systems (FRSs). The interface layer allows an application to be independent from the idiosyncrasies of specific FRS software and enables the development of generic tools that operate on many FRSs.

- Ontology web language

  The Ontology Web Language (OWL), from the World Wide Web Consortium (W3C), is the most recent development in standard ontology languages (Bechhofer et al., 2004). OWL makes it possible to describe concepts but it also provides new facilities. It has a richer set of operators - e.g. intersection, union and negation. It is based on a different logical model, which makes it possible for concepts to be defined as well as described. Complex concepts, can therefore be built-up in definitions out of simpler concepts. Furthermore, the logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognize which concepts fit under which definitions. The reasoner can therefore help to maintain the hierarchy correctly. This is particularly useful when dealing with cases where classes can have more than one parent. There are three types of OWL, and are the following:

  - OWL-Lite

    OWL-Lite is the syntactically simplest sub-language. It is intended to be used in situations where only a simple class hierarchy and simple constraints are needed. For example, it is envisaged that OWL-Lite will provide a quick migration path for existing thesauri and other conceptually simple hierarchies.

  - OWL-DL

    OWL-DL is much more expressive than OWL-Lite. OWL-DL and OWL-Lite are based on Description Logics (hence the suffix DL, described in section 2.3.6). Description Logics are a decidable fragment of First Order Logic and are therefore amenable to automated reasoning. It is therefore possible to automatically compute the classification hierarchy and check for inconsistencies in an ontology that conforms to OWL-DL.

  - OWL-Full

    OWL-Full is the most expressive OWL sub-language. It is intended to be used in situations where very high expressiveness is more important than being able to guarantee the decidability or computational completeness of the language. It is therefore not possible to perform automated reasoning on OWL-Full ontologies.

2.3.6 Applications and projects

Several applications can be found in the literature related to the ontology construction and deployment. Therefore, the main contributions in connection with the manufacturing process.

The Ontology Inference Layer or Ontology Interchange Language (Fensel, 2003) known as OIL, is a proposal for a joint standard for specifying and exchanging ontologies on the Web. OIL is entirely web-driven and is based on:

1. Description logic (DL), which provides formal, clean and well defined semantics and efficient reasoning support. DL is part of a research effort in knowledge representation to provide theories and systems for expressing structured knowledge, and for accessing and reasoning with it in a principled way.

2. Frame-based systems, which provide epistemologically rich modeling primitives. OIL incorporates the essential modeling primitives of Ontology Exchange Language (XOL) into its language. OIL is based on the notion of concept and the definition of its super-classes and attributes. Relations can also be defined not as attributes of a class, but as independent entities having a certain domain and range. Like classes, relations can be arranged in a hierarchy.

   While in DL roles are not defined for concepts (actually, concepts are defined as subclasses of restricted role), in a frame context a class is a subclass of its attribute definitions (i.e., all instances of the class must fulfill the restrictions defined for the attributes). Asking which roles could be applied to a class does not make much sense in DL, as nearly all slots can be applied to a class, while with frame-based modelling the implicit assumption made is that only those attributes which are defined for a class can be applied to that class. The ontology definitions encoded by XOL include both schema information, such as class definitions, and non-schema information, such as object definitions.

3. Existing standards such as Open Knowledge Base Connectivity (OKBC), and new Internet standards such as Extensible Markup Language (XML) and Resource Description Framework (RDF) which provide syntactically exchangeable notations. OIL is intended to improve OKBC, XML and RDF with necessary features for expressing rich ontologies and its core language has been designed so that it provides most of the modelling primitives commonly used in frame-based ontologies and automated reasoning support (e.g., class consistency and assumption & subsumption checking). OIL shares many features with OKBC and defines a clear semantics and XML-oriented syntax for them (extending OKBC). In the same way as OIL provides an extension of OKBC (and is therefore downward compatible with it), OIL provides an extension of XML and RDF.

   Techniques for performance evaluation, developed for XML, can directly be used for ontologies specified in OIL because the XML syntax of OIL is defined by using the Extensible Markup Language Streams (XMLS) mechanism. XMLS incorporates the notion of inheritance and this allows to capture the semantics of the "is-a" relationship. RDF and RDF-Schema (RDFS) are further candidates for a Web-based syntax for OIL. The relationship between OIL and RDFS is much closer than that between OIL and XMLS. This is not surprising, since XMLS was meant to generalize the way of defining the structure of valid XML documents, and RDFS was meant to capture meaning in the way semantic nets do. In the same way as RDFS is used to define itself it can also be used to define other ontology languages. Therefore, a syntax for OIL is defined by giving an RDFS for the core of OIL and an extension to this RDFS is proposed to complement this core by covering further aspects.

Even if OIL is based on DL, verification and validation in OIL/XML are basically syntax-based, whereas in logic they are typically based on theorems. It is not yet clear how the OIL/XML/RDF community will deal with soundness and completeness.

Nonetheless, OIL proposers believe that the existing Ontolingua design for an ontology interchange language is not appropriate as a standard ontology language for the Internet and put forward OIL as an alternative and better standard.

2.3.7 Ontology editors

There are a number of more or less generic editors to create and manage ontologies. Some of them are presented next.

- The Stanford ontolingua ontology editor Standford-University (2010) was, in 1999, the most standard editor to create ontologies. It is a Web-based tool for creating, editing and browsing ontologies in the Ontolingua language.

- OilEd is a simple ontology editor developed by Bechhofer et al. (2001) at the University of Manchester. OilEd allows the user to: i) build ontologies; ii) use the FaCT reasoner to check the consistency of ontologies and add implicit subClassOf relations; iii) export ontologies in a number of formats including both OIL-RDF and DAML-RDF. The intention behind OilEd is to provide a simple, freeware editor that demonstrates the use of, and stimulates interest in, OIL.

  OilEd is not intended as a full ontology development environment. It does not actively support the development of large-scale ontologies, the migration and integration of ontologies and many other activities that are involved in ontology construction. Rather, it offers just enough functionality to allow users to build ontologies and to demonstrate how the FaCT reasoner can be used to check and enrich ontologies. To get the full benefit from OilEd, it is also necessary to have the CORBA-FaCT reasoner installed. The latest Windows version also includes the FaCT reasoner.

- WebOnto (Domingue, 2010) (HC-REMA, PATMAN and Enrich projects) is an on-line tool for collaborative construction of ontologies. In addition to implementing the standard HTTP protocol, the LispWeb server offers a library of high-level Lisp functions to dynamically generate HTML pages, a facility for dynamically creating image maps, and a server-to-server communication method. WebOnto enables knowledge engineers to browse and edit ontologies represented in the knowledge modelling language OCML.

- Protègè-2000 is a tool for ontology editing and knowledge acquisition. Protègè-2000 has hundreds of users who use it for projects ranging from modelling cancer-protocol guidelines to modelling nuclear-power stations. Protègè-2000 is aimed at making it easier for knowledge engineers and domain experts to perform knowledge-management tasks. One of the major advantages of the Protègè-2000 architecture is that the system is constructed in an open source, modular fashion. Its component-based architecture enables system builders to add new functionalities to Protègè-2000 by creating appropriate plug-ins such as support for alternative storage formats and domain-specific user-interface components. From Protègè, it is possible to export ontologies to other knowledge-representation systems, such as RDF, OIL and DAML (Pagels, 2006).

- OntoEdit (Sure et al., 2002) is an off-line tool which enables developing, inspecting and modifying ontologies. It uses a GUI to codify conceptual structures (concepts, concept hierarchy, relations and axioms). Ontologies in OIL format can be imported and it is possible to export ontologies in OIL and Frame-Logic formats.

2.3.8 Ontology reasoning

Reasoning and querying over data type properties are important and necessary tasks if these properties are to be understood by machines.

For instance, e-shops may need to classify items according to their sizes, and to reason that an item which has height less than 5 cm and the sum of length and width less than 10 cm belongs to a class, called "small-items", for which no shipping costs are charged. Accordingly the billing system will charge no shipping fees for all the instances of the "small-items" class.

Various ontology languages, such as Resource Description Framework (RDF), OIL, DAML+OIL (Gómez-Pérez and Suárez-Figueroa, 2003) and Ontology Web Languages, have witnessed the importance of data-types in the Semantic Web. All of them support data-types. For instance, the DAML+OIL language supports unary data type predicates and qualified number restrictions on unary data type predicates, e.g. a "less than 21" predicate could be used with the data type property age to describe objects having age less than 21. Description Logics (DLs), a family of logical formalisms for the representation of and reasoning about conceptual knowledge, are of crucial importance to the development of the Semantic Web. Their role is to provide formal underpinnings and automated reasoning services for Semantic Web ontology languages such as OIL, DAML+OIL and OWL.

Using data types within Semantic Web ontology languages presents new requirements for DL reasoning services. Firstly, such reasoning services should be compatible with the XML Schema type system, and may need to support many different datatypes. Furthermore, they should be easy to extend when new data types are required.

An important aspect of future systems exploiting these resources is the ability to process OWL (Web Ontology Language) documents and OWL Knowledge Bases (OWL-KBs), which is the official semantic web ontology language. Ontologies may be taken or extended for domain-specific purposes (domain-specific ontologies extend core ontologies). For doing this, a reasoning system is required as part of the ontology editing system. For example, the utilization of a reasoner, such as RacerPro, can process OWL Lite as well as OWL DL documents (knowledge bases). Some restrictions apply, however. OWL DL documents are processed with approximations for nominal in class expressions and user-defined XML data-types are not yet supported.

The reasoners provide with the following services to OWL ontologies and RDF data descriptions:

- Check the consistency of an OWL ontology and a set of data descriptions.
- Find implicit subclass relationships induced by the declaration in the ontology.
- Find synonyms for resources (either classes or instance names).
- Provide extensional information from OWL documents (OWL instances and their relationships), which is needed for client's applications.
- HTTP client for retrieving imported resources from the web. Thus, multiple resources can be imported into one ontology.
- Incremental query answering for information retrieval tasks (retrieve the next n results of a query).

2.4 Decision support systems

The concept of DSS was introduced, from a theoretical point of view, in the late 1960s. Klein and Methlie (1995) define a DSS as a computer information system that provides information in a specific problem domain using analytical decision models as well as techniques and access to databases, in order to support a decision maker in taking decisions effectively in complex and ill-structured problems. Thus, the basic goal of a DSS is to provide the necessary information to the decision maker, in order to help him to get a better understanding of the decision environment and the alternatives he faces.

Decision-making in general, and specifically financial decision-making, has been significantly improved, in the last two decades, through the rapid progress of information technology and computer science. Decision-making in the financial management field is a very complicated process, where decision makers (managers of companies, managers of credit institutions, individual investors, engineers, etc.) face, on a daily basis, a large volume of information that should be examined in order to make the final decision concerning the performance or the viability of a firm, the granting or denying of a credit application, the construction and management of a portfolio, the choice of an investment, or the construction of a financial marketing plan, etc.

Typically, the phases of the decision-making process overlap and blend together, with frequent looping back to earlier stages as more is learned about the problem, as solutions fail, and so forth. Figure 2.5 describes what probably comes to be a more customarily used model of the decision-making process in a DSS environment. A first step is the recognition of the problem or an opportunity. Once the problem recognized, it is defined as a term that facilitates the creation of the model. Some authors state that emphasis must be placed in the next two steps: the model development and the alternatives analysis. After that, the choice is made and implemented. As final step and if necessary, a new recognition should be done. Obviously, no decision process is this clear-cut in an ill-structured situation (Shim et al., 2002).

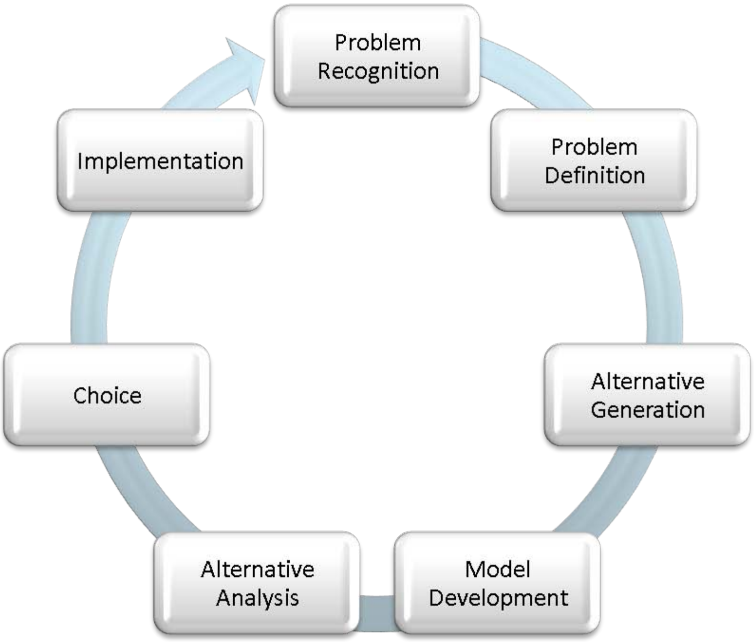


Figure 2.5: Decision-making process.

There has also been a huge effort in the DSS field for building a group support system (GSS) or Collaboration Support Systems in order to enhance the communication-related activities of team members engaged to computer- supported cooperative work. The communication and coordination activities of team members are facilitated by technologies that can be characterized along the three continua of time, space, and level of group support (Alavi and Keen, 1989; Shim et al., 2002). Teams can communicate synchronously or asynchronously; they may be located together or remotely; and the technology can provide task support primarily for the individual team member or for the group's activities. These technologies are utilized to overcome space and time constraints that burden face to face meetings, to increase the range and depth of information access, and to improve group task performance effectiveness, especially by overcoming "process losses".

The combination of decision theory with the new knowledge and the powerful tools offered by computer science and information technology, leads to the development of new types of information systems able to support decision makers and improve the decision-making process. Basically, the efforts in supporting the whole decision-making process focused on the development of computer information systems providing the support needed. Initially, two types of systems were developed: i) decision support systems (DSS) and ii) expert systems (ES). Although these two approaches were very promising, their implementation revealed several problems.

To overcome these problems without missing the advantages of both ESs and DSSs, a new type of intelligent system called a knowledge-based DSS (KBDSS) has been proposed. The basic characteristic of this new approach is the integration of ES technology with models and methods used in the decision support framework, such as mathematical programming methods, multicriteria decision aid methods, and multivariate statistical methods.

In the 21st century, internet, web and telecommunications technologies can be expected to result in organizational environments that will be increasingly more global, complex, and connected. Courtney (2001), following Mitroff and Linstone (1993), suggests that DSS researchers should embrace a much more comprehensive view of organizational decision-making (see figure 2.6) and develop decision support systems capable of handling much "softer" information and much broader concerns than the mathematical models and knowledge-based systems have been capable of handling in the past. This is an enormous challenge, it is imperative face if DSS is to remain a vital force in the future. The primary difference between figure 2.6 and typical decision models in a DSS context is the development of multiple and varied perspectives during the problem formulation phase. Mitroff and Linstone (1993) suggest that perspectives be developed from organizational (O), personal (P) and technical (T) positions. In addition, ethical and aesthetic factors are considered as well. The mental models of stakeholders with various perspectives lie at the heart of the decision process, from defining what a problem is, the analysis of the results of the problem.

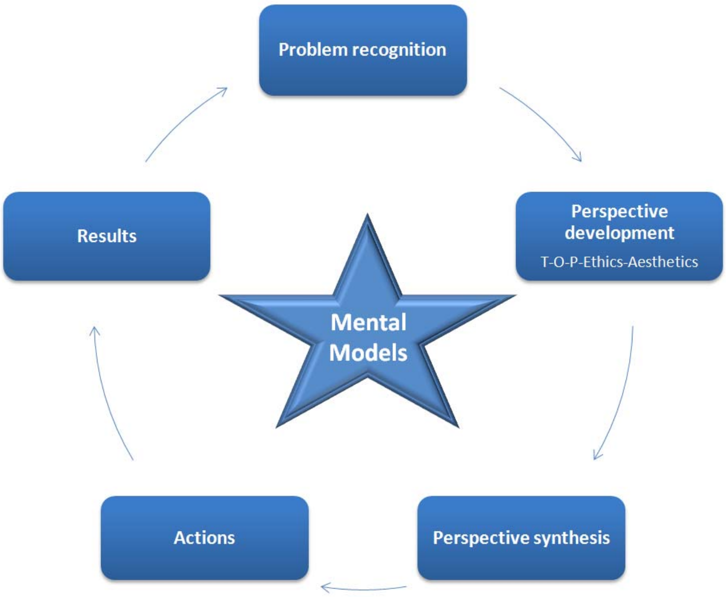


Figure 2.6: A new decision paradigm for DSS, source (Courtney, 2001).

Since the early 90's, four powerful tools have emerged for building DSS. The first new tool for decision support was the data warehouse. Next, on-line analytical processing (OLAP) and the data-mining appeared. Finally, the fourth new toolset has been the technology associated with the World Wide Web. Precisely, this last one is the center of activity in developing DSS. This refers to a computerized system that delivers decision support information or decision support tools to a manager or business analyst using a Web browser such as Netscape Navigator or Internet Explorer

(Power and Sharda, 2007).

2.4.1 Optimization-based decision support systems

Optimization-based decision support can be divided into three stages: formulation, solution, and analysis.

Formulation: it refers to the generation of a model in the form acceptable to a model solver. Converting a decision-maker's specification of a decision problem into an algebraic form and then into a form understandable by an algorithm is a key step in the use of a model. It has been a long way from the days of requiring an optimization problem to be input in the commonly used Mathematical Programming System (MPS) format. Several algebraic modeling language processor systems (AMLPS) have been developed that make it convenient to input the modeler's form of an optimization problem directly into a solver.

Solution: it refers to the algorithmic solution of the model.

Historically, most of the research effort in operations research (OR) has been concentrated on development of new algorithms to solve problems faster. Decision support software developers appear to incorporate advances in the solution algorithms quite quickly to let the user benefit from these enhancements. For example, the traditional linear programming software continues to be refined in both simplex method and interior point algorithms. The emphasis is on taking advantage of problem characteristics to reduce the problem size or to speed up a specific algorithmic step. The result is the ability to solve really large problems. It has also enabled the modelers to consider uncertainty in the decision situation through stochastic programming with recourse type approaches.

Analysis: it refers to the 'what-if' analysis and interpretation of a model solution or a set of solutions. Only recently have vendors of optimization software begun to focus on the final stage of the modeling process-analysis. This stage includes delivery of model solution in a usable form to enhance the ability to analyze and understand the problem and the solution. For instance, the report generating functionality is now a common feature used to present the results to the user in a usable form that can be integrated into databases.

Solutions can also be stored in popular spreadsheet formats for simple graphical analysis or report generation. Some modeling environments offer their own graphical display tools to display results in easy to use format. It is likely that the growth of new visualization tools will benefit the process of solution delivery in OR models as well. Thus, it would be possible to incorporate multimedia in highlighting solutions or especially exceptions to the norm or signal infeasibilities.

The development of DSS tools to support these three stages has occurred at different rates. Research in optimization traditionally focused on generating a better solution algorithm; as the technologies have evolved, more progress has been made in the formulation and analysis functions of DSS support.

2.4.2 Data and information

One of the most important contributions to the informatics data standards is the extensible markup language (XML), which describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them. XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language [ISO 8879]. By construction, XML documents are conforming SGML documents.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data is made up of characters, some of which form character data, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure.

XML was developed by an XML Working Group (originally known as the SGML Editorial Review Board) formed under the auspices of the World Wide Web Consortium (W3C) since 1996. It was chaired by Bosak (1997) with the active participation of an XML Special Interest Group (previously known as the SGML Working Group) also organized by the W3C.

The design goals for XML are:

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

This specification, together with associated standards (Unicode and ISO/IEC 10646 for characters, Internet BCP 47 and the language sub-tag registry for language identification tags), provides all the information necessary to understand XML Version 1.0 and construct computer programs to process it.

2.5 Thesis objectives

The thesis characterizes an interdisciplinary research on computer science techniques (rule-based reasoning, case-based reasoning, ontologies and planning) applied to manufacturing control systems, in the different chemical processes (continuous, batch and semi-continuous).

Also an important activity such as data integration could be carried out. Regarding the previous, applying semantic web technologies to enable data integration of disparate data sources based on the semantics (meaning) of the data, is a promising research line.

Particular objectives that will allow the culmination of the thesis (and also seen as opportunities for Ontology research areas) are:

- To create of a robust structure chemical process domain.
- To provide a controlled vocabulary.
- To provide the possibility to exploit the generalization/specialization of information.
- To check the consistency, validation and verification of the ontology domain.
- To allow communication between and among people and organizations (e.g. to unify different research fields by a common vocabulary).
- To facilitate the inter operability among systems, e.g. using the ontology as an interlingua to unify different languages and software tools.
- To allow the re-usability of the ontology, when represented in a formal language that can be (or become so by automatic translation) a re-usable and/or shared component in software systems and hardware agents.
- To allow the reliability: A formal representation facilitates automatic consistency checking.
- And to develop some libraries of the ontology for future sharing.

Chapter 3

Methods and Tools

In this chapter, the background of those methods and tools that are utilized in the development and implementation of the different models presented throughout this thesis are described. In general, methodologies give a set of guidelines of how you should carry out the activities identified in the ontology development process, what kinds of techniques are the most appropriate in each activity and what are the results obtained as described in Section 3.1. Such methodologies provided the basis for building and improve methodology, which is described in Section 3.2. Finally, it is necessary to describe the ontology layering and its architecture which establishes the level of complexity that has been adopted in the present ontological development approach.

3.1 Toward a Methodology

Currently, a variety of ontologies are being developed and used in diverse research areas. Each development project usually follows its own set of principles, in order to design criteria and phases in the ontology unfolding process, as described in Section 2.3.3. The absence of standard guidelines and methods hinders the following aspects:

- The development of shared and concentrated ontologies within and between projects.
- The extension of a given ontology by others.
- Its reuse in other ontologies and final applications.

It is widely recognized that constructing ontologies, or domain models, is an important step in the development of knowledge-based systems (KBS). However, there is a lack of consensus on a uniform approach to design and maintain these ontologies. Various methodologies exist to guide the theoretical approach that is taken, and numerous ontology building tools are available. The problem is that these procedures have not coalesced into popular development styles or protocols, and the tools have not yet matured to the degree one would expect in other software instances. Consequently. two key methodologies have been deeply studied to guide the development of this thesis work.

Methontology is a structured method to build ontologies. It is based on the experience acquired in developing an ontology in the domain of chemicals.

It enables experts and ontology makers who are unfamiliar with implementation environments to build ontologies from scratch. Methontology identifies the following activities in the development of an ontology: specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, documentation. The life cycle of the ontology is based on the refinement of a prototype and ends with a maintenance state. The most distinctive aspect of Methontology is the focus on this maintenance stage (Figure 3.1).

Specification: The goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. The formality of the ontology specification document varies on its degree of formality depending on if you use natural language, competency questions or a middle-out approach. Uschold and Gruninger (1996) give excellent argument on the use of a middle-out as opposed the classic bottom-up and top-down approach in identifying the main terms of your glossary. The main advantage of the middle-out approach is that it allows you to identify the primary concepts of the ontology you are starting on. After reaching agreement on such terms and their definition you can move on to specialize or generalize them, only if they are necessary.

Knowledge Acquisition: It is concerned about acquiring the knowledge from sources like experts, books, handbooks, figures, tables and even other ontologies are sources of knowledge from which it can he elucidated using in conjunction techniques such as brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools.

Conceptualization: In this phase a complete Glossary of Terms (GT) must be built. Terms include concepts, instances, verbs and properties. So, the GT identifies and gathers all the useful and potentially usable domain knowledge and its meanings.

This concepts must be described as Fernandez-Lopez et al. (1997) suggest: Data Dictionary, which describes and gathers all the useful and potentially usable domain concepts, their meanings, attributes, instances, etc.; Tables of Instance Attributes, which provide information about the attribute or about its values at the instance; Tables of Class Attributes, to describe the concept itself not its instances; Tables of Constants, used to specify information related to the domain of knowledge that always takes the same value; Tables of Instances, where instances are defined; and attributes classification trees, to graphically display attributes and constants related in the inference sequence of the root attributes, as well as the sequence of the root attributes, as well as the sequence of formulas or rules to be executed to infer such attributes.

Integration: The reuse of definitions already built into other ontologies instead of starting from scratch should be consider. Methontology proposes the development of an integration document, summarizing: the meta-ontology to be used. Although for each and every term whose definition is going to be used: the name of the term in the conceptual model, the name of the ontology from which you will take its definition, the name of the definition and its arguments in the ontology.

Implementation: Ontologies implementation requires the use of an environment that supports the meta-ontology and ontoiogies selected at the integration phase. The result of this phase is the ontology codified in a formal language such us: CLASSIC, BACK, LOOM, Ontolingua, Prolog, C++ or any informatics language.

Evaluation: Evaluation means to carry out a technical judgment of the ontologies, their software environment and documentation with respect to a frame of reference during each phase and between phases of their life cycle. Evaluation subsumes the terms Verification and Validation.

- Verification refers to the technical process that guarantees the correctness of an ontology, its associated software environments, and documentation with respect to a frame of reference during each phase and between phases of its life cycle.
- Validation guarantees that the ontologies, the software environment and documentation correspond to the system that they are supposed to represent.

Documentation: Methontology intends to break this circle including the documentation as an activity to be done during the whole ontology development process. In fact, some documents also must be done after some phase as: after the specification phase, a requirements specification document; after the knowledge acquisition phase, a knowledge acquisition document and finally, after the conceptualization, a conceptual model document that describes the application domain.

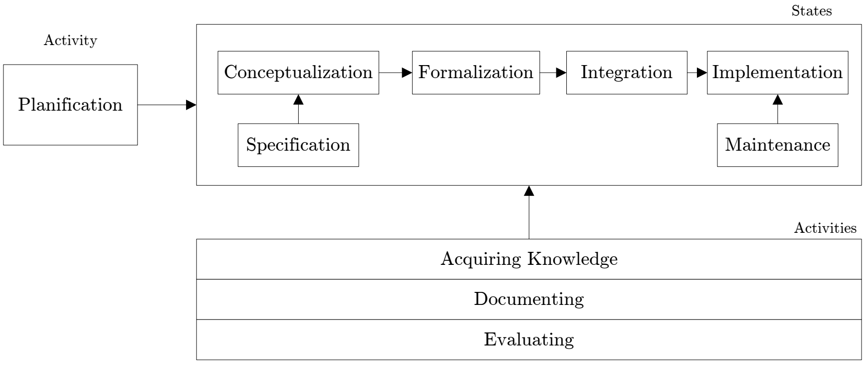


Figure 3.1: Methodology of methontology.

On-to-Knowledge methodology includes the identification of goals that should be achieved by knowledge management tools and is based on an analysis of usage scenarios. The steps proposed by On-To-Knowledge are: (i) kick-off, in which some competency questions are identified, potentially reusable ontologies are studied and a first draft of the ontology is built; (ii) refinement, in which a mature and application-oriented ontology is produced; (iii) evaluation, in which requirements and competency questions are checked and the ontology is tested in the application environment; and finally (iv) ontology maintenance. On-To-Knowledge stresses that the ontology modeling process should start with a definition of the abstraction level, which is strongly dependent on the usage of the ontology (Figure 3.2).

The Institute AIFB (Germany) provides a methodology that includes guidelines for introducing ontology-based knowledge management concepts and tools into enterprises, helping knowledge providers and seekers to present knowledge efficiently and effectively. The methodology captures lessons learned from the On-To-Knowledge case studies while applying the On-To-Knowledge tool set.

In this way On-To-Knowledge is based on two main features or process that must be developed: The knowledge meta process and the knowledge process.

The knowledge meta process consists of five main steps. Each step has numerous substeps, requires a main decision to be taken at the end and results in a specific outcome. The main stream indicates steps (phases) that finally lead to an ontology-based KM application. The phases are: Feasibility Study, Kickoff, Refinement, Evaluation and Application & Evolution.

- Feasibility Study: This step contemplates a feasibility study, e.g. to identify problem/opportunity areas and potential solutions. In general, a feasibility study serves as a decision support for economical, technical and project feasibility, determining the most promising focus area and target solution.
- Kickoff: The outcome of this phase is a semi-formal description of the ontology. It refers to the creation of an ontology requirements specification document. It describes what an ontology should support, sketching the planned area of the ontology application and listing and should guide an ontology engineer to decide about inclusion and exclusion of concepts and relations and the hierarchical structure of the ontology.
- Refinement: Approaches for modeling should be considered as top-down, middle-out and bottom-up. Top-down approach refers to modeling concepts and relationships from a very generic level to specific one. Middle-out approach refers to identify the most important concepts, which will then be used to obtain the remainder of the hierarchy trough generalization and specialization. And finally bottom-up approach, where relevant concepts are extracted semi-automatically from available documents.

  To formalize the above mentioned, a taxonomy comes out of the semi-formal description of the ontology and add relations other than the "is-a" relation, which forms the taxonomic structure. The ontology engineer adds different types of relations as analyzed, e.g. in the competency questions to the taxonomic hierarchy. The outcome of this phase is the "target ontology", that needs to be evaluated in the next step.

- Evaluation: In this phase, three tasks of evaluation are distinguished: (i) technology-focused evaluation, which consists of two main aspects, the evaluation of properties of ontologies generated by development tools and the evaluation of the technology properties; (ii) user-focused evaluation which evaluates whether an ontology-based application is at least as good as already existing applications that solve similar tasks, and; (iii) ontology-focused evaluation which leads to more correct hierarchies of ontologies. The outcome of this phase is an evaluated ontology, ready for the roll-out into a productive system.
- Application & Evolution: In this step the usage of ontology-based systems is being described by the knowledge process. Also there are some rules in the evolution or updates of the ontology that implies testing all possible effects to the application. Most important is therefore to clarify who is responsible for maintenance and how it is performed and in which time intervals is the ontology maintained.

The knowledge process. Once a Knowledge application has been fully implemented in an organization, knowledge processes circle around the following steps:

- Knowledge creation and/or import of documents and meta data.
- Knowledge items have to be captured in order to elucidate importance or inter-linkage.
- The retrieval of and access to knowledge.
- The use of knowledge in an specific context.

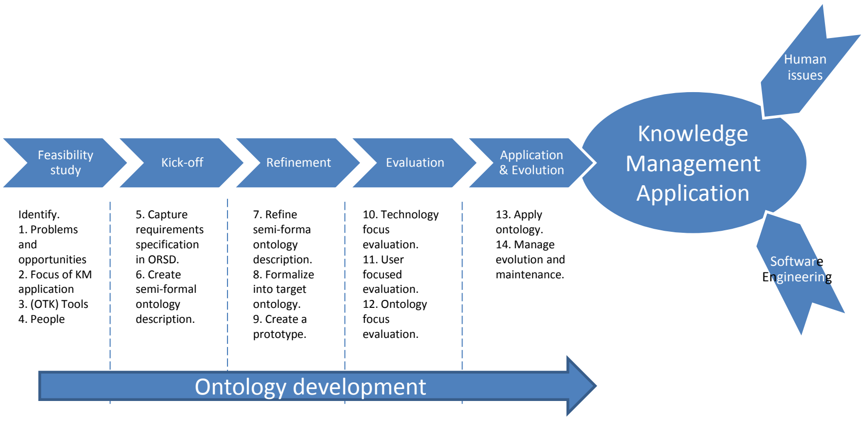

Feasibility study
Kick-off
Refinement
Evaluation
Application & Evolution
Knowledge Management Application
Human issues
Software Engineering
Identify.
1. Problems and opportunities
2. Focus of KM application
3. (OTK) Tools
4. People
5. Capture requirements specification in ORSD.
6. Create semi-formal ontology description.
7. Refine semi-forma ontology description.
8. Formalize into target ontology.
9. Create a prototype.
10. Technology focus evaluation.
11. User focused evaluation.
12. Ontology focus evaluation.
13. Apply ontology.
14. Manage evolution and maintenance.
Ontology development

Figure 3.2: Methodology of On-To-Knowledge.

3.2 Proposed methodology

The methodology used in this thesis work is based on two ontology development methodologies "Methontology" (López et al., 1999) and "On-To-Knowledge" (Sure and Studer, 2002). The aforementioned methodologies have been inserted into a Plan, Do, Check/Study and Act Cycle (PDCA or PDSA), which results in an ordered sequence of steps, that are easy to understand and track (Figure 3.3).

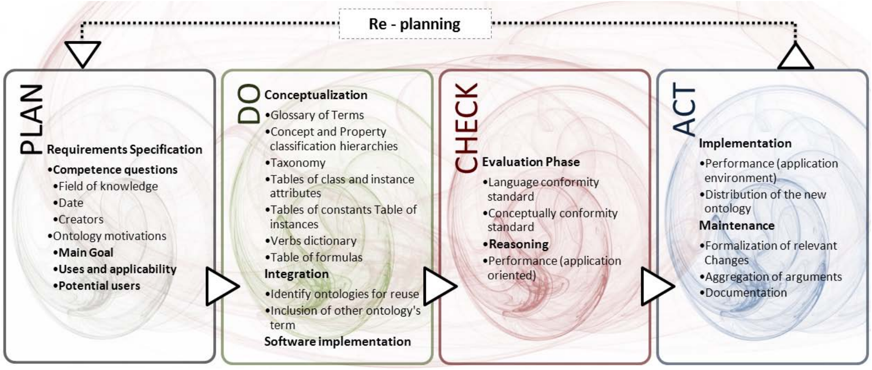

Re - planning
PLAN
Requirements Specification
•Competence questions
•Field of knowledge
•Date
•Creators
•Ontology motivations
•Main Goal
•Uses and applicability
•Potential users
DO
Conceptualization
•Glossary of Terms
•Concept and Property classification hierarchies
•Taxonomy
•Tables of class and instance attributes
•Tables of constants Table of instances
•Verbs dictionary
•Table of formulas
Integration
•Identify ontologies for reuse
•Inclusion of other ontology's term
Software implementation
CHECK
Evaluation Phase
•Language conformity standard
•Conceptually conformity standard
•Reasoning
•Performance (application oriented)
ACT
Implementation
•Performance (application environment)
•Distribution of the new ontology
Maintenance
•Formalization of relevant Changes
•Aggregation of arguments
•Documentation

Figure 3.3: Methodology for developing enterprise ontology.

PDSA Cycle is a four step cycle for problem solving, which includes: planning (definition of a problem and a hypothesis about possible causes and solutions), doing (implementing), checking (evaluating the results), and action (back to plan if the results are unsatisfactory or standardization if the results are satisfactory). The PDSA cycle emphasizes the prevention of error recurrence by establishing standards and the ongoing modification of those standards. Even before the PDCA cycle is employed, it is essential that the current standards be stabilized. The process of stabilization is often called the SDCA (standardize-do-check-action) cycle.

The PDSA cycle has been changed since its beginnings at 1900, being known as the Shewhart Cycle, the Deming Cycle, the PDCA Cycle, and the PDSA Cycle. When Deming (1993) reintroduces the Shewhart cycle in 1986 stated that "Any step may need guidance of statistical methodology for economy, speed, and protection from faulty conclusions from failure to test and measure the effects of interactions".

The planning step of the improvement cycle required prediction and associated theory. The third step compared the observed data to the prediction as a basis for learning. The improvement of this tool came with the use of the word "study" in the third phase of the cycle, which emphasizes that the purpose of this phase is to build new knowledge.

The PDSA cycle (Figure 3.4) is applicable to all types of organizations and to all groups and levels in an organization, and some important issues attained are well known as:

- It provides a framework for the application of improvement methods and tools guided by theory of knowledge:
  - It encourages planning to be based on theory.
  - Its theory leads to appropriate questions which provide the basis for learning.

- Its questions lead to predictions which guide the user in identifying the necessary data, methods and tools to answer the questions relative to the theory in use.
- It emphasizes and encourages the iterative learning process of deductive and inductive learning.

- It allows project plans to adapt as learning occurs.
- It provides a simple way for people to empower themselves to take action that leads to useful results in the pragmatic tradition of learning.
- It facilitates the use of teamwork to make improvements.

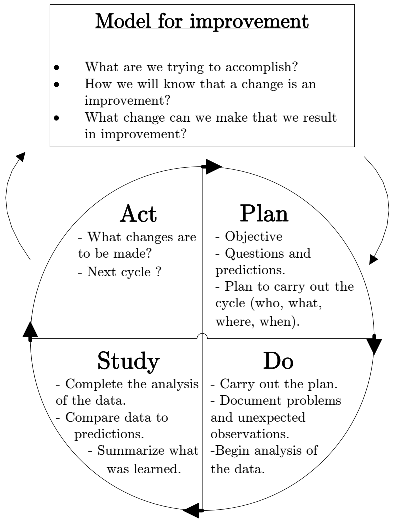

Model for improvement
• What are we trying to accomplish?
• How we will know that a change is an improvement?
• What change can we make that we result in improvement?
Act
- What changes are to be made?
- Next cycle ?
Plan
- Objective
- Questions and predictions.
- Plan to carry out the cycle (who, what, where, when).
Study
- Complete the analysis of the data.
- Compare data to predictions.
- Summarize what was learned.
Do
- Carry out the plan.
- Document problems and unexpected observations.
-Begin analysis of the data.

Figure 3.4: Model for improvement of the PDSA cycle.

Regarding the PDSA cycle, it is worth mentioning that spending the right time in each phase or activity of the PDSA cycle is imperative to having a smooth and meaningful quality improvement process. Cycle elements involve a reflective process based on the scientific method, and ensure improvement efforts are carried out by the success to be achieved.

3.2.1 Plan phase

The purpose of this phase is to investigate the current situation, fully understand the nature of any problem to be solved, and to develop potential solutions to the problem. Once the problem is understood a specification document must be produced, which can be modified later, but the main idea is to guide the process development and establish the main targets. This methodology proposes to apply the three main knowledge acquisition approaches in the presented order. The order of application, that follows below, has been extensively tested rendering the most satisfactory results.

- Requirements specification: This step determines important information that characterizes the project itself, such as people involved in the project and other important information, e.g. due dates.
  - Project name (ontology name)
  - Date
  - Creator
  - Other information
- Domain definition: It is important to state the sources of knowledge that will be used. As was aforementioned, the knowledge acquisition is proposed in a specific order. In this way, an order of source knowledge is proposed. Thus, the use of official documents like standards, taxonomies and glossaries of terms is a first option covering the middle-out approach. Then, the use of books and handbooks for improving the bottom-up approach. Finally, the use of expertise from experts, to cover the Top-down approach is suggested.
  - Knowledge sources: A key step is the definition of the source of domain knowledge (Specialized Dictionaries in the domain, Handbooks of the domain, Review books, articles about the domain).
  - Knowledge acquisition approaches
    1. Middle-out approach: Refers to identify the most important concepts which will then be used to obtain the remainder of the hierarchy by generalization and specialization.
    2. Bottom-up approach: Refers to modeling relevant concepts extracted semi-automatically from available documents.
    3. Top-down approach: Refers to modeling concepts and relationships from a very generic level to specific one.
- Competence questions (Ontology motivations): The main ideas of the ontology to be developed must be established. In this step the ontology scope, its main purpose, its use and possible applicability are defined as well as its potential users
  - Field of knowledge
  - Main Goal
  - Analysis requirements
  - Language definition
  - Uses and applicability
  - Potential users

3.2.2 Do phase

The purpose of this phase is to implement the work toward ontology building with help of the previously planned. The details of this plan should include the necessary aspects to begin the development of the ontology, e.g. what data will be collected, who collects the data, the time line, etc.

- Domain Conceptualization
  - Glossary of terms: The first step is to build a glossary of terms. The glossary of terms identifies and collects all those useful and potential, as well as its correct meaning for the domain which ontology is trying to modeling. In addition, the glossary of terms will also help extracting relations between concepts (properties), attributes, as well as guiding the building of rules and axioms.
  - Taxonomy: In order to formalize the above mentioned terms, a taxonomy of the terms must be done. It is important to be sure that the "is-a" relation, from bottom to up hierarchical structure, is accomplished.

    An affinity diagram is a useful tool helping to develop this activity of structuring the taxonomy in a both creative and logical manner (Figure 3.5). There are six basic steps to creating an affinity diagram:

    1. Identify the problem or issue: Applied to the taxonomic structure, the concepts are the issues identified previously.
    2. Each person writes issues related to problem on note cards or sticky notes: Applied to the taxonomic structure, the main concepts (The ones found at the top of the hierarchy) must be identified or created.
    3. Organize the cards or sticky notes into logical piles: Applied to the taxonomic structure, all the concepts must be organized following the "is-a" relation.
    4. Name each pile with a header: Applied to the taxonomic structure, it must be decided whether there are some more general concept upon the top (from the hierarchy) concepts.
    5. Draw an affinity diagram: Applied to the taxonomic structure, in some way the structure must be saved, with the pertinent notes if there were any.
    6. Discuss the piles created: Applied to the taxonomic structure, our taxonomy must be reviewed and none "is-a" relation can be violated.

    Finally the concepts will be named later classes of the ontology.
  - Table of verbs: The most usual verbs used in the domain can help to find rapidly the properties in our domain.
  - Table of properties: There are two types of properties. On the one hand, there are properties that relate different concepts of our domain, called object-type properties. On the other hand, there are the properties which describe relationships between individuals and data values, called Data-type properties.

    For the object-type properties, a very useful tool for structuring the development of the properties, is the interrelationship matrix. This is a square matrix where all the concepts are placed in both axis (X & Y) as headers. At each intersection between concepts, a relation must be defined if it exists. This process must be run two times, the first one is X/Y in top-down direction and the second is Y/X in left-right direction.
  - Table of restrictions (axioms): The restrictions build constraints between the properties above mentioned. Some examples of restrictions are quantifier restrictions, cardinality restrictions and hasValue restrictions.

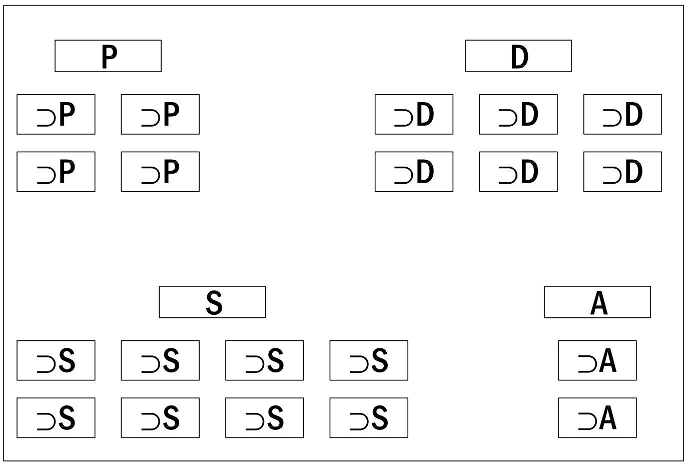


Figure 3.5: Affinity diagram tool.

  - Tables of instances: A table of instances or individuals must be done. The individuals represent objects in the domain.

- Formalization (implementation of the ontology editor): This step should be done if a platform that provides a set of tools to build the domain model is needed. Actions to create, view and manipulate the model will result in structures in the domain of knowledge-modeling. The capture of classes, object properties, data properties, axioms, rules, cases, etc., should be implemented in an ontology editor. The selection of the editor must define the main ontology language and the possible formats to be imported.

- Integration: Finally, the search of other ontological models already developed is suggested. It is important to emphasize that the ontologies that can be taking in to account, must belong to the same domain. For the development of this final step, two activities should be covered.

  - Identify ontologies for reuse
  - Inclusion of other ontology's term

- Informatics implementation: In order to finish this phase, a light implementation in any informatics language (e.g. C, C++, Java) should be developed. This code must use the ontological model, for a first evaluation of the behavior in the specific domain, and also for the particular tasks that the ontology was designed.

3.2.3 Check/Study phase

In this phase five steps are suggested to realize.

- Language conformity standard: As a first step looking for possible syntaxes language errors within the ontological model must be done. In this sense the language must be in agreement with the informatics standards of language, it means to be in accordance to the rules from the world wide web consortium (W3C), which is an international community that develops standards that ensure the growth of the Web over time.

- Conceptually conformity standard: The second step concerns to all the concepts that belong to the domain and model our ontology. Those classes now must have a description concept in agreement with any standard of the domain. Also the help of experts is necessary to ensure the generality of the model.
- Reasoning: The third step concerns to the reasoning which is the process that checks the consistency of the ontological model. In addition, the reasoning process performs the identification of the domain and range of properties (relationships) among the classes that compose the ontology.
- Performance (application oriented): A complete informatics application is designed and implemented at this stage. Based on any informatics language such as C, C++ or Java, a framework capable of exploiting the ontological model should be developed. Therefore the ontological model or semantic model serves as the base model (main structure) for the specific purpose, or the main ones, that the ontology was designed for (this part is an expansion of the previous light informatics implementation).
- Revise: Finally the analysis of the effect of the application in the domain must be done. Basically, it consists of comparing the new data to the baseline data in order to determine whether an improvement is achieved, and whether the measures in the aim statement are met. Such analysis can be performed by means of: Pareto charts, histograms, run charts, scatter plots, control charts or radar charts. As a result, two tasks are suggested to be implemented to accomplish the revision phase:
    1. Reflect on the analysis, and consider any additional information that emerged as well. Compare the results of the test against the measurable objective.
    2. Document lessons learned, knowledge gained, and any surprising results that emerged.

3.2.4 Act phase

The objective of this step is twofold: on the one hand to ensure the correct implementation, and on the other hand to carry out the operations related to the ontology maintenance.

- Implementation
    - Performance (application environment): After the creation of the complete informatics application, its behavior in the actual domain environment should be tested based on potential users experience.
    - Distribution of the new ontology : Once the ontology application has been fully implemented in the actual domain environment, it is necessary to distribute it to the final users. Thus, a users manual must be created containing documents and data about the use of the ontological application. Apart from this, the necessary training in using the new tool should be given.

- Maintenance
  - Formalization of relevant changes: The addition of arguments about possible changes within the ontology domain regarding the addition of new relations between concepts or the modification of these relations must be explained.
  - Documentation of important changes: Along the use of the implementations, reasons for important changes may arise. Therefore all these reasons must be documented in order to keep track off the relevant changes.

3.2.5 Re-planning phase

Toward a useful re-planning of the ontological model and computer application, the following methodology is proposed to apply a SWOT analysis (Strengths, Weaknesses, Opportunities and Threats) over the life of the ontological structure (the ontological model + computer application).

- Domain overview (addition of new concepts and relationships)
- Ontology overview
- Develop ontology changes
- Informatics application overview

After posing the previous questions, the PDSA cycle should be restarted.

This simple and powerful process allows continuous efforts to achieve important improvements regarding the efficiency, effectiveness and performance of the ontological framework.

3.3 Ontology layering and architecture

The ontology architecture is classified according to the factors that influence its complexity, such as concepts, taxonomy, patterns, constraints and instances, as shown in Table 3.1.

Table 3.1: Rating for the conceptualization complexity of ontologies

Rating	**Conceptual Model Criteria**
Very low	Only concepts
Low	Taxonomy, high number of patterns, no constraints
Nominal	Taxonomy with general patterns for properties available, some constraints
High	Taxonomy with properties and axioms, few modeling patterns, considerable number of constraints
Very High	Taxonomy with properties, axioms and instances, no patterns, considerable number of constraints

Furthermore Figure 3.6 introduces the six layers that are seen as the main axis of the ontology architecture.

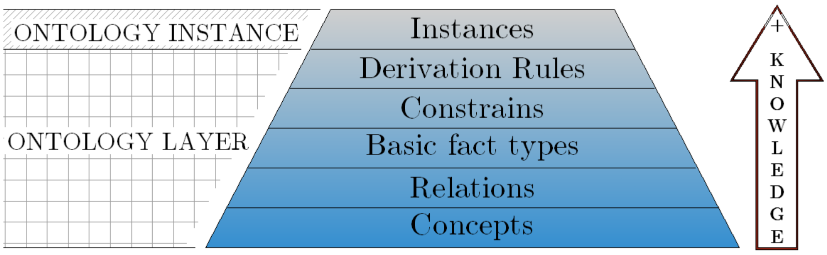


Figure 3.6: Construct ontology layering

Such layers represent the following features:

Concepts: A concept defines a basic and abstract idea that is commonly used in an ontology domain. It is represented as a word or phrase.

Relations: A relation describes the way in which two or more concepts are interrelated. It is usually described by a verb or verb phrase (basic properties).

Basic fact types: A basic fact type is a kind of primitive sentence or fact. It is composed of concepts and relations. If the basic fact type is always true in the ontology that contains it, it can play a role as an axiom in the logic-based ontology (detailed properties).

Constraints: A constraint is the restriction that is applied to a fact type (binary or numerical restrictions for properties).

Derivation rules: These are rules, functions or operators (including mathematical calculations or logical inference) that are used to derive new facts from existing ones.

Instances: Instances have the particularities of processing and are specifications in the applications of the upper layers.

Chapter 4

An ontology-based for Chemical Process Engineering: Enterprise Ontology

4.1 Introduction

From the previous discussion in section 1.1, it is clear the need for infrastructures continuously and coherently support for fast and reliable decision-making activities related to the production process, is now of paramount importance (Venkatasubramanian et al., 2006). This need is more evident when we consider recent activity in the fields of data warehousing, online analytical processing (OLAP), data mining and Web-based DSS, followed by the treatment of collaborative support systems and optimization-based decision support (Shim et al., 2002). It is quite common for process activities to have large databases. Hence, an enormous amount of information is created, stored and shared and it may be hard to find the right information when it is required. Furthermore, because of the possible use of different computer languages and differences in conceptualization, the interoperability between information in different systems is one of the most critical aspects in the daily operation of many organizations.

Indeed, shrinking profits have made it essential to exploit large databases (as companies need to manufacture many non-cyclical products with complex recipes, etc.) using non-generic/blind methods. One key aspect is information extraction, which should result in the extraction of information quality. Information quality can be defined as precise information in terms of time, content, and clarity (Eppler, 2006). A common problem is that this data extraction process may be performed using blind methods in many cases. However, the performance of such methods can be drastically improved by combining them with knowledge or expertise of the process.

Information is data that is processed to be useful. Knowledge is the application of data and information through the development of a system which models, in a structured way, the experience gained in some domain. Knowledge exists as soon as human interaction is or has been made available in any step of the product/process development (Gebus and Leiviskä, 2009). In recent years, there has been an effort to create knowledge with a minimum human interface, either in a straight and formal way (e.g. expert systems) or in a conceptual manner. The use of multiple models to represent detailed and abstract knowledge of chemical processes has been taken into account recently. In particular, this knowledge representation enables us to identify process sections together with their function, objectives and relations within the process. This enables the automatic generation of alternative views of the process, organized in a hierarchy of different levels of abstraction.

The way to address these problems is to reduce or eliminate conceptual and terminological confusion and come to a shared understanding. Such an understanding can function as a unifying framework as an ontology which can be adopted to develop an integrated framework, through the definition and semantic description of data and information. This is the basis for modeling the different forms of knowledge that are to be organized or the contextualized information that can be used to produce new meanings and generate new information.

As explained in section 2.3 ontologies constitute a means of specifying the structure of a domain of knowledge in a generic way that can be read by a computer (formal specification) and presented in a human-readable form (informal specification). Moreover, ontologies are emerging as a key solution to knowledge sharing in a cooperative business environment (Missikoff and Taglino, 2002). Since they can express knowledge (and the relationships in the knowledge) with clear semantics, they are expected to play an important role in forthcoming information-management solutions to improve the information search process (Gruber, 1993; Obrst, 2003). This chapter develops the ontology steps following the methodology described in section 3.2.

4.2 Plan phase

As proposed in section 3.2, the three main knowledge acquisition approaches have been applied as follows.

4.2.1 Requirements specification

The lack of integration among the enterprise hierarchical levels does not allow complete optimization of companies' functions. Information is needed from different hierarchical levels when an important change is necessary. However, the desired change cannot be made unless the system is robust. In addition, the need to integrate the different modeling approaches in a hierarchical decision-support system means that consistent terminology and concepts must be used to improve the communication and collaboration tasks over the entire system.

The problem of process management has been studied in many works (Rippin, 1983; Subrahmanyam et al., 1995). These works have made a rich and vast description of such problems and have pointed out open issues where research efforts may provide significant improvements, in which the in-depth description aids understanding.

- Project name (ontology name): The name of the project is *Integrated Enterprise Ontology Project.*
- Date: Period ranging from December 2007 to December 2011.
- Creator: Mr. Edrisi Muñoz Mata.
- Other information: –

4.2.2 Domain definition

The scope of this ontology lies in the process systems engineering (PSE) domain. Specifically, it is of utmost importance to coordinate and integrate information and decisions among the various functions that comprise the whole supply chain. Recently, enterprise-wide optimization (EWO) has emerged as a new area which aims at optimizing the operations of supply, production and distribution to reduce costs and inventories. Specifically, EWO places emphasis on production facilities focusing on their planning, scheduling and control taking into account the knowledge in the area of chemical engineering. In this area, only some modest attempts at integrating a small subset of enterprise-wide decision models exist, since the complex organizational structures underlying business processes challenge our understanding of cross-functional coordination and its business impact (Varma et al., 2007). Models and tools that allow a comprehensive application of the EWO are a research field that has not been deeply studied yet. Therefore, the model has been constructed maintaining the coherence, based on the needs to solve optimization problems. Thus, the ontology has to include a level of detail sufficient enough to describe all the necessary and sufficient problem features, but at the same time it has to be general enough to represent any EWO problem.

In order to define this field, for its future modeling, the following three key areas have been considered: supply chain, planning and scheduling, and process control activities.

The supply chain (SC) concept can be defined as the group of interlinked resources and activities required to create and deliver products and services to customers. In this area decisions are taken at different stages within the supply chain and at different levels in the management hierarchy. SC performance is a result of the synchronization of materials, information and cash flows along the SC elements, such as suppliers, manufacturers, distributors and retailers. The feasibility of the management of the alternatives depends on several restrictions such as mass balances, capacity constraints, technological constraints, budgeting limitations and customer satisfaction, among others. The main SC drivers relate to economic performance defining the SC profitability. The decisions involved in this area are concerned to those associated with to the strategical decision level.

Planning and scheduling comprise the organization of human and technological resources in a company within a range of days to weeks in order to directly satisfy customer demands defined by a production plan resulting from the company planning function (Korovessi and Linninger, 2006). The planning and scheduling function usually involves deciding on the amount of products, the allocation of resources to the needed tasks, the order in which the different batches are to be performed and at what time these tasks are to be started. This problem is stated as an optimization problem that seeks for minimizing the production makespan, lateness, earliness, or any performance function that could be adopted. The production process itself is defined in the production recipe, which contains the stages that are to be performed in different units, and the operations that are to be carried at each stage. In addition, the constraints regarding operational timing such as simultaneity or sequential must be described. Other issues, such as the material balance, the resource consumption and processing times are also described in the production recipes.

Process control aims at achieving an efficient, safe, environmentally friendly and reliable operation to execute the production requests calculated at the scheduling level. Thus, the real time execution in the plant and the optimization of dynamic trajectories of process variables are addressed. Specifically, the problem of batch process management has been studied in many works (Rippin, 1983; Subrahmanyam et al., 1995).

Process control performs several functionalities in batch processes management. On the one hand, the batch operation model should be implemented in a feed-forward manner. This model is defined in the named control recipes, obtained from the optimization solution of the scheduling system and the master recipes which have been optimized in a previous process design step. Control recipes are composed by a serial of process operations, phases and the transition logics to be implemented in a specific equipment piece. Additionally, processing conditions, actuation variables and set points for each stage are given in the control recipes.

On the other hand, basic control is carried out. Firstly, automatic feedback control is used to reduce the influence of uncertainty and reject intra-batch disturbances, since in batch processing it is crucial to satisfy quality specifications in each batch. In addition, supervision may be performed to improve processing conditions from run to run and this constitutes a recipe adjustment. Process monitoring and fault diagnosis are also employed to handle faults or exceptions and perform manual actions, if required. Finally, interlock is used to provide safety or to avoid mistakes in processing the batch. As for the performance objectives in process control level, there can be from either economic (for example, maximize profit or profitability) or processing type (for example, to accomplish customer specifications, minimize product variability, meet safety or environmental regulations or maximize the plant flexibility with reliable control).

- Knowledge sources: As a source for the domain knowledge for the PSE, EWO, SC, planning and scheduling, and process control, the following documents are considered.

  For the process control part, the ANSI/ISA-88 standard is studied. ANSI /ISA-88 standard (International Society for Measurement and Control, 1995, 2006, 2007a; Shirasuna, 2007)), allows to create a general infrastructure to be applied to any process system. The ANSI/ISA-88 representation provides an advantage of establishing a more general conceptualization in the batch process domain. Such a generalization is behind years of joint work by recognized batch manufacturing experts who met to define a perceptive view of batch plants organization and its corresponding hierarchy of control functions. As a consequence, following the ANSI/ISA-88, virtually all activities concerning batch processes can be properly represented.

  The ANSI/ISA-88 standard is composed of five parts:

  - ANSI/ISA-88.01-1995 Batch Control Part 1: Models and terminology.
  - ANSI/ISA-88.00.02-2001 Batch Control Part 2: Data structures and guidelines for languages.
  - ANSI/ISA-88.00.03-2003 Batch Control Part 3: General and site recipe models and representation.
  - ANSI/ISA-88.00.04-2006 Batch Control Part 4: Batch production records.

– ISA-TR88.00.02-2008 Machine and Unit States: An implementation example of ISA-88.

The basic relations can be taken from the assertions of the models that ANSI/ISA-88 describes. These assertions show the basic relations between physical model, recipe model and procedural model and are described by entity-relationship diagrams (E-R diagrams). For ANSI/ISA-88, recipes are needed because of the set of information that uniquely identifies the production requirements for a specific product. Figure 4.1(a) shows the relations between recipes types. Some models are also described, such as the procedural control model shown in Figure 4.1(b), which describes the relation that occurs in a process cell. The process model shown in Figure 4.1(c), which is very conceptual and describes the functionality needed to create a batch, describes the steps that must occur to make a product.

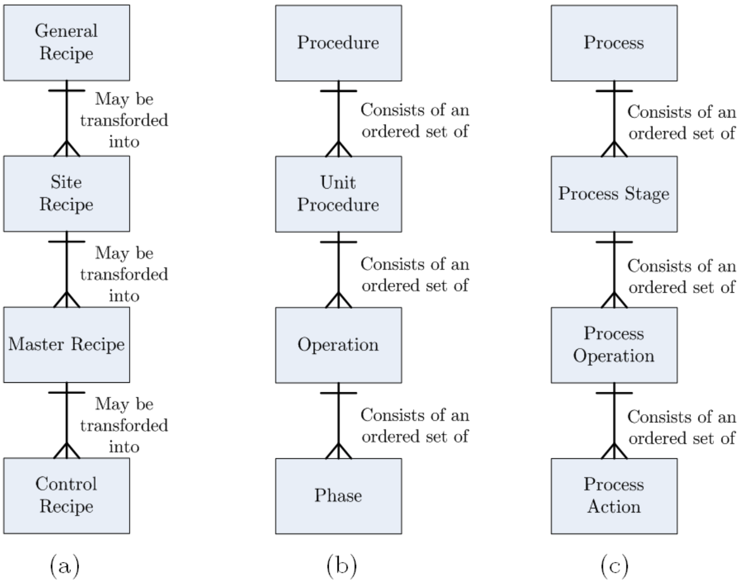


Figure 4.1: ANSI/ISA-88 recipe, procedure and process models.

All these relations must involve the control model, process model and the physical model to accomplish process functionality. Figure 4.3 shows some basic relations of the aforementioned models.

For the planning and scheduling area, the use of the ANSI/ISA-88&95 standards allows a real integration and enables future control related events to be taken into account at scheduling level, in order to minimize the occurrence of critical situations during the execution of the process. It also provides a definition for any required information from the process measurements that are made by the control and fault diagnosis system.

Concepts of functions (order processing, detailed production scheduling, production control, quality assurance, etc.) can thus be developed according to the ANSI/ISA-95 standard, which contains five parts:

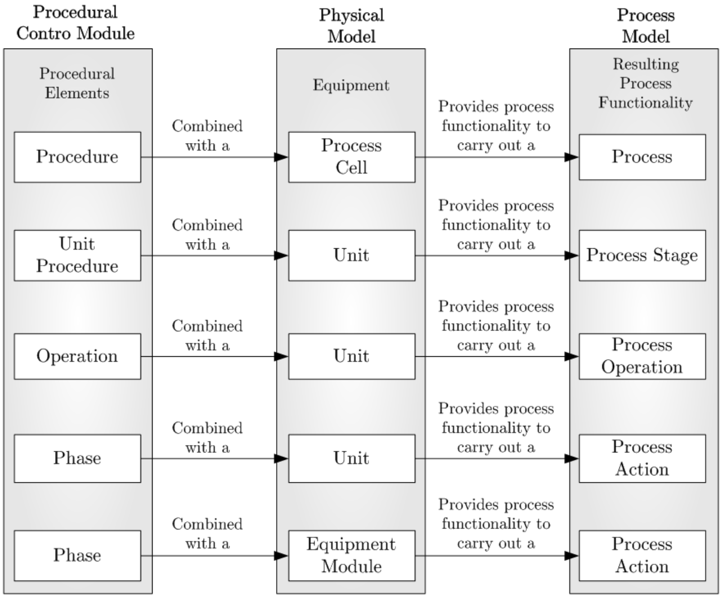


Figure 4.2: Basic relations between models in ANSI/ISA-88.

- ANSI/ISA-95.00.01-2000, Enterprise-Control System Integration Part 1: Models and Terminology
- ANSI/ISA-95.00.02-2001, Enterprise-Control System Integration Part 2: Object Model Attributes
- ANSI/ISA-95.00.03-2005, Enterprise-Control System Integration, Part 3: Models of Manufacturing Operations Management
- ISA-95.04 Object Models & Attributes Part 4 of ISA-95: Object models and attributes for Manufacturing Operations Management
- ISA-95.05 B2M Transactions Part 5 of ISA-95: Business to manufacturing transactions

The ANSI/ISA-95 has been also used for the supply chain areas, since it stands for the integration of enterprise and control systems, and contains the terminology and models that can be used to determine which information, has to be exchanged between systems for sales, finance and logistics and systems for production, maintenance and quality. The ANSI/ISA-95 standard can be used for several purposes, for example as a guide for the definition of user requirements, for the selection of manufacturing execution system (MES) suppliers and as a basis for the development of MES systems and databases. Besides, SC handbooks (Chopra and Meindl, 2004; Raleigh and Harmelink, 2004) and SC reviews (Grubic and Fan, 2010) have been revised for further accuracy in the domain definition.

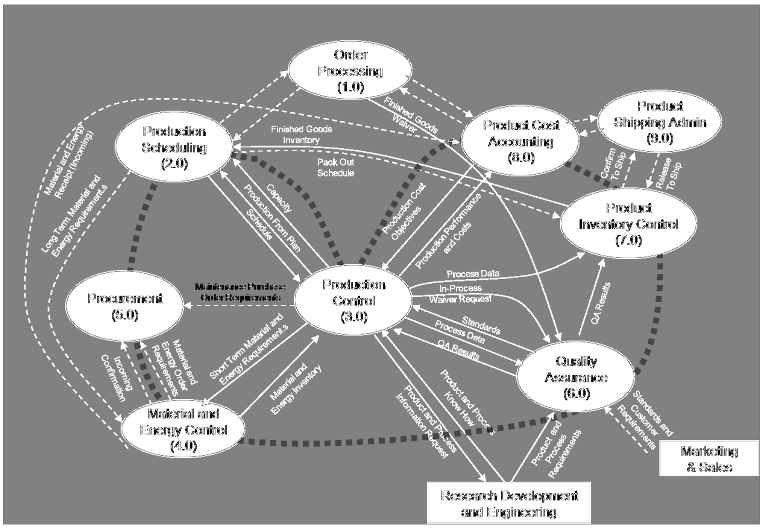

Order Processing (1.0)
Production Scheduling (2.0)
Production Control (3.0)
Material and Energy Control (4.0)
Procurement (5.0)
Quality Assurance (6.0)
Product Inventory Control (7.0)
Product Cost Accounting (8.0)
Product Shipping Admin (9.0)
Research Development and Engineering
Marketing & Sales
Material and Energy Receipt (Incoming)
Long Term Material and Energy Requirement.s
Finished Goods Inventory
Pack Out Schedule
Finished Goods Waiver
Capacity
Production From Plan
Schedule
Production Cost Objectives
Production Performance and Costs
Confirm To Ship
Release To Ship
Maintenance Purchase Order Requirements
Process Data
In-Process Waiver Request
Standards
Process Data
QA Results
QA Results
Short Term Material and Energy Requirement.s
Material and Energy Inventory
Incoming Confirmation
Material and Energy Order Requirements
Product and Process Information Request
Product and Process Know How
Product and Process Requirements
Standards and Customer Requirements

Figure 4.3: Basic relations between models in ANSI/ISA-95.

4.2.3 Competence questions (ontology motivations)

In order to accomplish the competence questions, the following information must be fulfilled:

1. Field of knowledge: Chemical process industry

2. Main Goal: The creation of an ontological framework capable of integrating the different hierarchical levels within the enterprise structure.

3. Analysis requirements:

   Next, the particular elements involved in ontology development comprising the enterprise, plan, recipe and control levels are described.

   Enterprise requirement Enterprise requirements that a knowledge representation system should meet are natural representation of the physical reality/-model (items of equipment/devices), automatic generation of abstraction levels to identify which sections of the process can be potentially improved, and the implementation the Purdue Reference Model, which is a detailed collection of the functional requirements of the generic information management and control to run a batch manufacturing process (Williams, 1989). Such a reference model can be hierarchically arranged in a large-scale computer system that integrates plant management, plant production scheduling, inventory management, individual process optimization, and unit process control for all of the plant's operating units as a whole. The reference model defines requirements that are common to all implementations, but it is independent of the specified requirements of any particular implementation that are applicable to existing processes.

Plant requirement Effective production is very important in today's global competitive environment. In the case of the batch process industry, multi-product and multipurpose plants, as well as continuous or semi-continuous processes, manufacture a variety of products through a sequence of operations that share available resources, intermediate products, and raw materials. The efficient use of such resources can be analyzed at different levels:

- If we focus on the planning area and control level, a process system involves multiple and interrelated activities that are performed at single or multiple sites, with different durations and amounts of information. Generally, information flows from the marketing department to the manufacturing department, which determines the production schedule that is needed to meet the sales strategies. In its most general form, the scheduling problem requires information that is related to the configuration of the plant (the available equipment units and resources), the product recipes (the set of processing tasks and resources required to manufacture a given product), precedence relationships between materials and final product requirements (demands and related due dates) (International Society for Measurement and Control, 1995).

  At this level, a closed-loop framework is proposed for the decision-making task of batch chemical plants. This framework integrates both scheduling and control and is based on ANSI/ISA-88 standards. Integration enables future events to be taken into account at scheduling level, in order to minimize the occurrence of critical situations during the execution of the process. It also provides any required information from the process measurements that are made by the control and fault diagnosis system. On the other hand, the opportunity for reactive scheduling allows the process to respond under unexpected schedule deviations or abnormal events.
- In a batch plant production environment, the occurrence of unpredictable events is usually unavoidable. Such events may be related to external market factors or to the intrinsic plant operation, and include equipment breakdowns and variable operation times. Despite the uncertainty in the environment, the scheduler has to make some decisions in order to start production and to face uncertainty when an abnormal event occurs.
- In addition, the integration of a control and monitoring system into process management helps to provide the process state information opportunely at different levels in the decision-making hierarchical structure, thus reducing the risk of incidents and improving the efficiency of the reactive scheduling by updating the schedules in the most effective way, which improves the process yield.

Unexpected events or disruptions can change the system status and affect its performance. Deviations from the original schedule and information about equipment breakdowns that is provided by the control and monitoring system will eventually trigger rescheduling. However, the schedule that is generated will be assessed according to the new plant situation. Thus, if some modifications are made, the newly created schedule will be translated into some control recipes for the actual process. Consequently, ontology can also be used to ensure the robustness of the running plan in the system.

The rescheduling system allows different dispatching rules, optimizers and objective functions to be selected, according to the process knowledge. Alternative rescheduling techniques (recalculate a new robust schedule, update operation times, reassignment, etc.) are evaluated and a system should select the most suitable ones, according to the objective function that is adopted. Optimization algorithms may be included, depending on the interest of the decision maker and the required reaction time.

Recipe requirements This integration approach follows the ANSI/ISA-88 batch control standard, which differentiates between four types of recipes: general, site, master and control. The general and site recipes are general recipes that are outside the scope of the control system. At control level, the information recipes are the master and control recipes.

Master recipes are derived from site recipes and are targeted at the process cell. A master recipe is a required recipe level; without it, control recipes cannot be created and batches cannot be produced. Master recipes take into consideration the equipment requirements within a given process cell. They include the following information categories: header, formula, equipment requirements and procedure. Control recipes are batches that are created from master recipes. They contain the product-specific process information that is required to manufacture a particular batch of product. They also provide the detail needed to initiate and monitor equipment procedural entities in a process cell.

Control activity requirements The control activity model International Society for Measurement and Control (1995) (Figure 4.4) provides an overall perspective of batch control and shows the main relationships between the various control activities. The control activities define how equipment in the batch manufacturing plant will be controlled.

Using this approach, recipes can be modified without changing the code of the PLCS and DCS that run the basic regulatory control. Moreover, recipes can run on different sets of equipment. The use of the ANSI/ISA-88 standard significantly reduces costs and implementation time through:

- Effective utilization and optimization of plant equipment, which maximizes total plant production capabilities.
- Reductions in total validation costs and production down-time via separate validation of recipe procedures and equipment.

4. Language definition: As seen in section 2.3.5 different ontology languages provide diverse facilities. Any language used to codify ontology- underpinned knowledge should be expressive, declarative, portable, domain independent and semantically well defined. The language used in an ontology is essential for its future implementation and sharing.

   We adopted OWL (Web Ontology Language), as it has good characteristics for ontologies (Bechhofer et al., 2004). OWL has been designed for use by applications that need to process the content of information, instead of just presenting the information to humans. OWL facilitates greater machine interpretability of

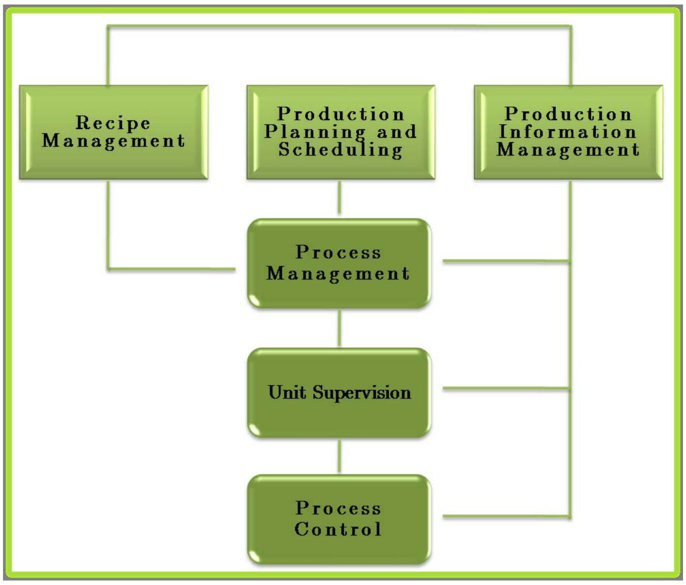

Recipe Management
Production Planning and Scheduling
Production Information Management
Process Management
Unit Supervision
Process Control

Figure 4.4: Control activity model

Web content than that supported by XML (Extensible Markup Language), RDF (Resource Description Framework), and RDF-S (Resource Description Framework Schema), as it provides additional vocabulary along with formal semantics. The ontology formally describes the meaning of the terminology used in documents. If machines are expected to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema.

OWL has been designed to meet this need for a Web Ontology Language, and is part of the growing stack of W3C recommendations that are related to the Semantic Web:

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents (XML-Core-Working-Group, 2009).
- XML Schema is a language that restricts the structure of XML documents and extends XML with data types (McQueen and Thompson, 2000).
- RDF is a data model for objects ("resources") and relations between them. It provides simple semantics for data models, which can be represented in XML syntax (Klyne and Carroll, 2002).
- Schema is a vocabulary for describing properties and classes of RDF resources. It includes semantics for generalization hierarchies of these properties and classes (Brickley and Guha, 2002).

OWL adds more vocabulary for describing properties and classes, including the relations between classes (e.g. disjointedness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

The semantics in the ontology build on XML's ability to define customized tagging schemes and RDF's flexible approach to representing data. This unifying aspect makes it easier to establish, through collaboration and consensus, the utilitarian vocabularies (between ontologies) needed for far-flung cooperative and integrative applications using the Word Wide Web and internal servers. The uses of these languages are helpful for the first task of the ontology, which is to become a standard tool for vocabulary, format, and definitions. Restrictions and reasoning make communication possible between the different system elements.

5. Uses and applicability: The ontological framework should be used for helping in the decision support-task, as a tool for extracting information quality as required. The framework will also be used as a quantitative support tool improving the use of optimization models.

6. Potential users: Any enterprise in which the automation has been implemented (formal specification). Also the enterprise decision makers related to strategic, tactical and operational decision levels (informal specification).

4.3 Do phase

As mentioned in the previous section, the modeling is based on the ANSI/ISA standards. Now that the knowledge sources and the goals of the project have been defined, it is time to formalize the model through the construction of the ontology. This means the implementation of the work based on the previous plan phase.

4.3.1 Domain conceptualization

- Glossary of terms: Table 4.1 contains the terms related to the upper level (strategic level) within the supply chain as enterprise requirements. The glossary identifies the usable and potential terms, as well as, their right meaning for the domain of interest. Thus, Table 4.2 defines some of the concepts that are fully shown in Table B.1 the terminology associated with the planning, scheduling and process control, based on the ANSI/ISA-88 standard.

Table 4.1: Supply chain main concepts.

Name	Description
Capacity	The physical facilities, personnel and process available to meet the product or service needs of customers. Capacity generally refers to the maximum output or producing ability of a machine, a person, a process, a factory, a product, or a service.
Customer	In distribution, the Trading Partner or reseller. In Direct-to-Consumer, the end customer or user.
Direct cost	A cost that can be directly traced to a cost object since a direct or repeatable cause-and effect relationship exists. A direct cost uses a direct assignment or cost causal relationship to transfer costs.
Distribution Center (DC)	The warehouse facility which holds inventory from manufacturing pending distribution to the appropriate stores.
Distributor	A business that does not manufacture its own products, but purchases and resells these products. Such a business usually maintains a finished goods inventory.
Facility	Place in the supply chain network where product is stored, assembled or fabricated.

Finished Goods Inventory	Products completely manufactured, packaged, stored, and ready for distribution.
Geographic placement	Point of the supply chain in which a facility can be installed.
Indirect cost	A resource or activity cost that cannot be directly traced to a final cost object since no direct or repeatable cause-and-effect relationship exists. An indirect cost uses an assignment or allocation to transfer cost.
Inventory	Raw materials, work in process, finished goods and supplies required for creation of a company's goods and services; The number of units and/or value of the stock of goods held by a company.
Inventory cost	The cost of holding inventory incurred by the shippers' supply chain network.
Inventory Management	The process of ensuring the availability of products through inventory administration.
Location	Deciding where a company will locate its facilities.
Market Demand	In marketing, the total demand that would exist within a defined customer group in a given geographical area during a particular time period given a known marketing program.
Production Planning and Scheduling	The systems that enable creation of detailed optimized plans and schedules taking into account the resource, material, and dependency constraints to meet the deadlines.
Supplier	A provider of goods or services. A seller with whom the buyer does business, as opposed to vendor, which is a generic term referring to all sellers in the marketplace.
Transportation	It entails moving inventory from point to point in the supply chain
Transportation cost	The total amount paid to various carriers for transporting products to customers.
Transportation Mode	The method of transportation: land, sea, or air shipment.

- Taxonomy: The aforementioned terms are formalized in a taxonomic manner. Specifically, the whole domain is organized in nine top terms or classes, as shown in figure 4.5. Each top class unfolds their corresponding subclasses. Figures 4.11 to 4.13 detail the taxonomy related to each top class. It is important to mention that description logic (DL), automatically name to the root class as Thing (owl:Thing). Finally the concepts will be named later classes of the ontology.

Table 4.2: Some concepts from ANSI/ISA-88.

Name	**Description**
Allocation	A form of coordination control that assigns a resource to a batch or unit. An allocation can be for the entire resource or for portions of a resource.
Arbitration	A form of coordination control that determines how a resource should be allocated when there are more requests for the resource than can be accommodated at one time.
Area	A component of a batch manufacturing site that is identified by physical, geographical, or logical segmentation within the site. An area may contain process cells, units, equipment modules, and control modules.
Basic control	Control that is dedicated to establishing and maintaining a specific state of equipment or process condition. Basic control may include regulatory control, interlocking, monitoring, exception handling, and discrete or sequential control.
Batch	The material that is being produced or that has been produced by a single execution of a batch process. An entity that represents the production of a material at any point in the process. Batch means both the material made by and during the process and also an entity that represents the production of that material. Batch is used as an abstract contraction of the words "the production of a batch."
Batch control	Control activities and control functions that provide a means to process finite quantities of input materials by subjecting them to an ordered set of processing activities over a finite period of time using one or more pieces of equipment.

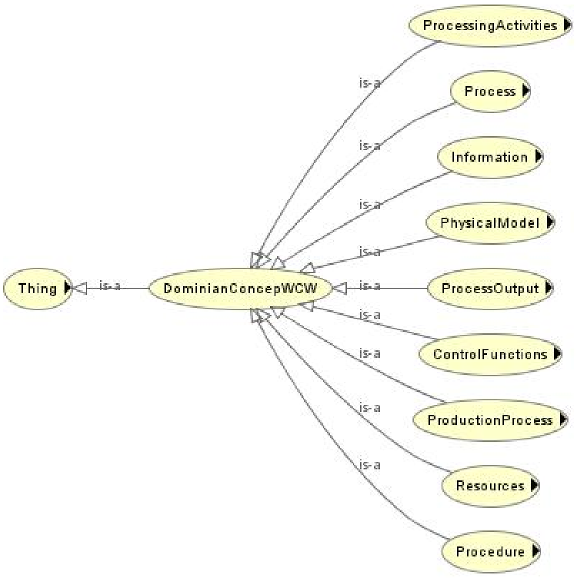


Figure 4.5: Taxonomy top classes (domain).

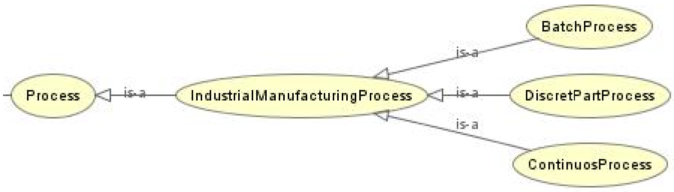


Figure 4.6: Process taxonomy.

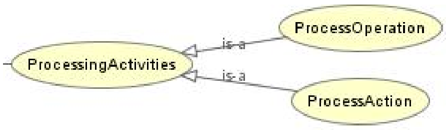


Figure 4.7: Processing activities taxonomy.

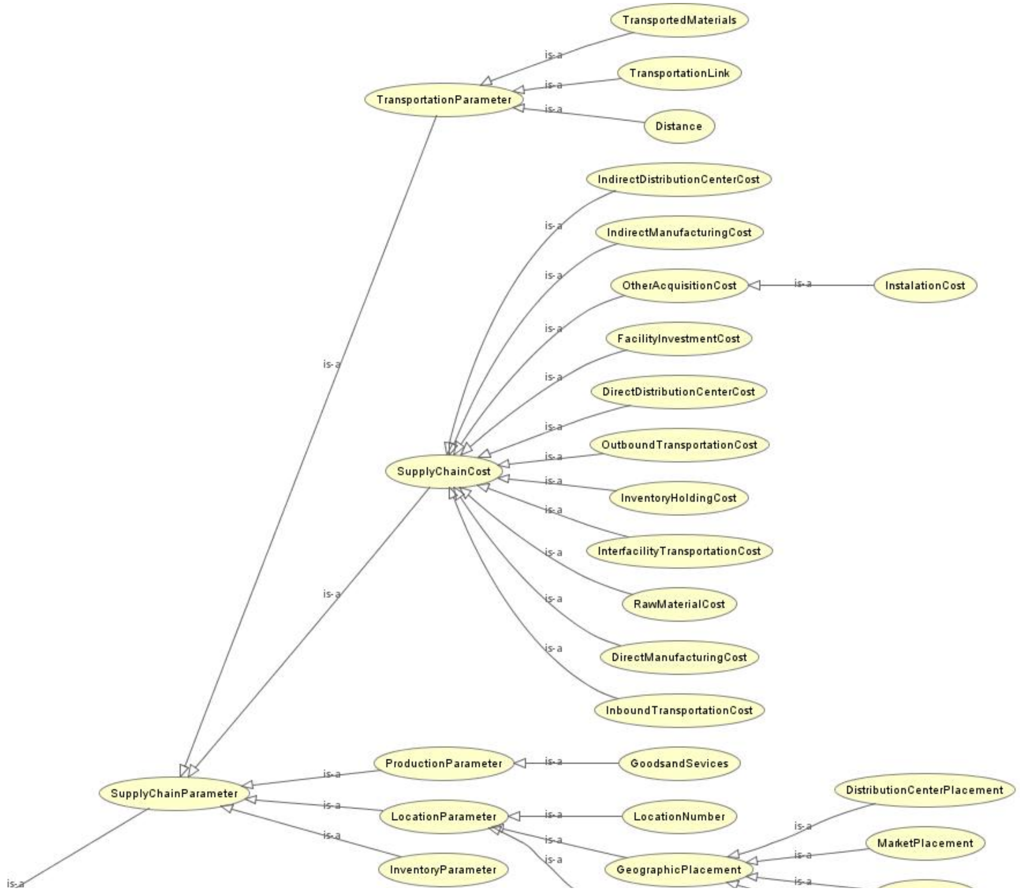


Figure 4.8: Information taxonomy (1/3).

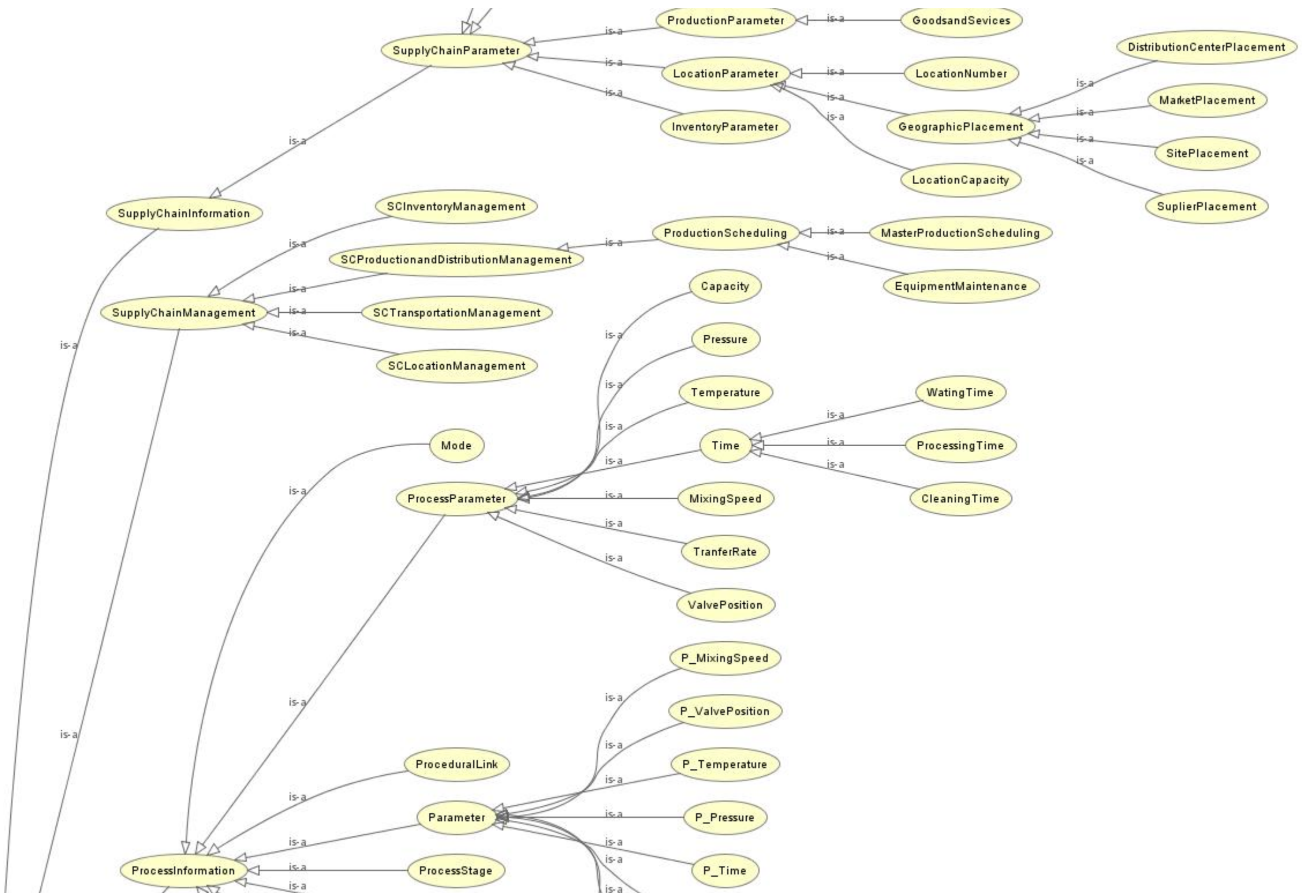


Figure 4.8: Information taxonomy (2/3).

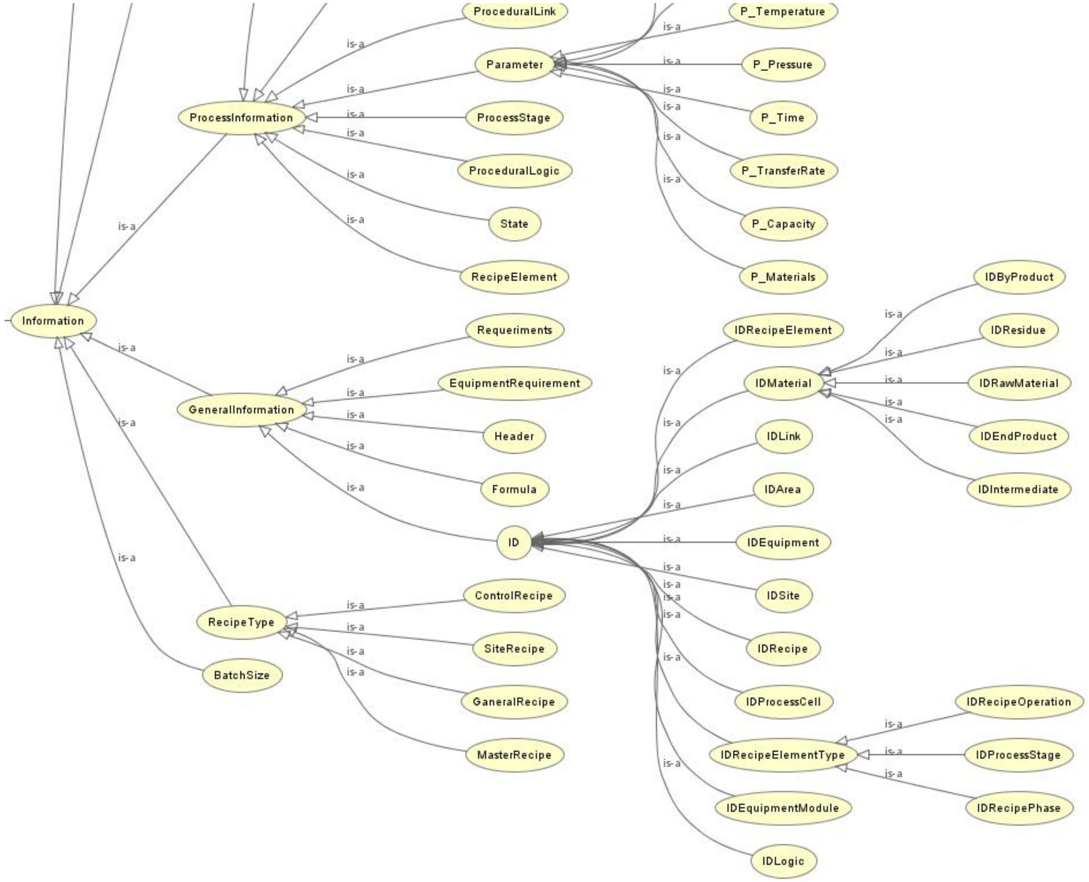


Figure 4.8: Information taxonomy (3/3).

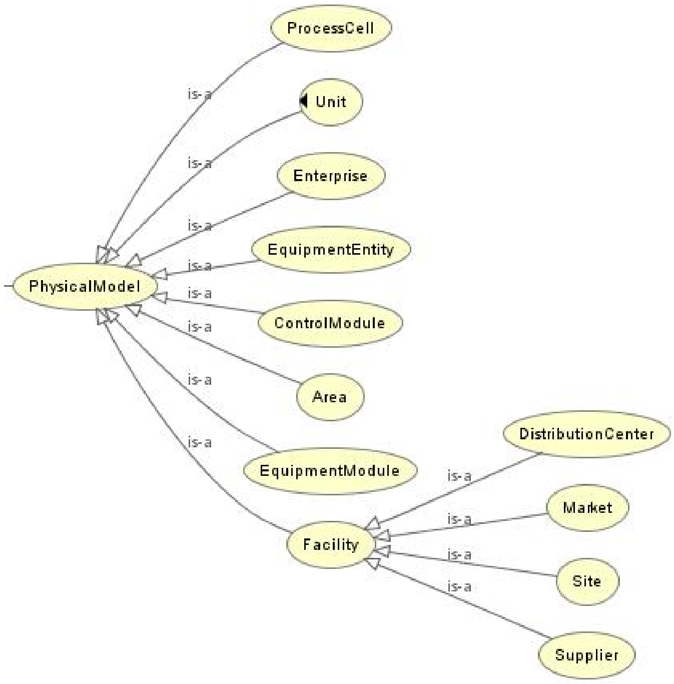


Figure 4.9: Physical model taxonomy.

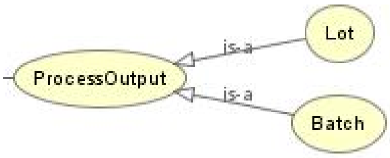


Figure 4.10: Process output taxonomy.

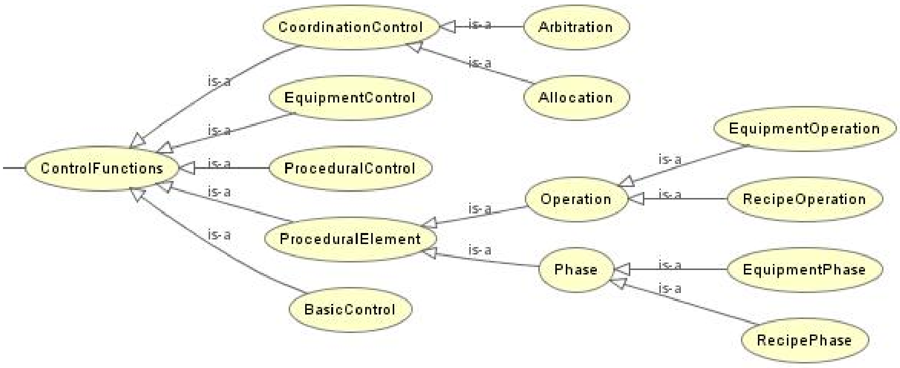


Figure 4.11: Control function taxonomy.

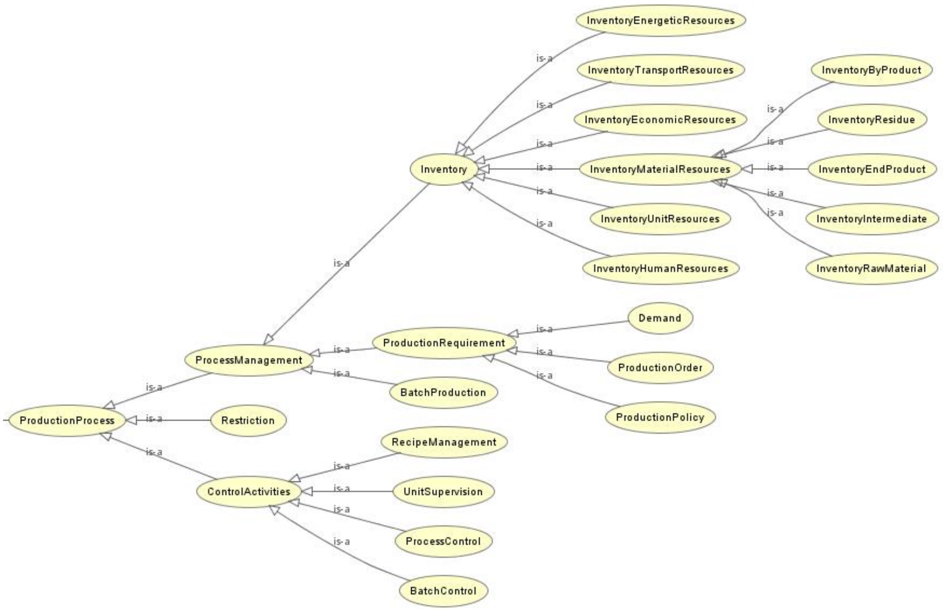


Figure 4.12: Production process taxonomy.

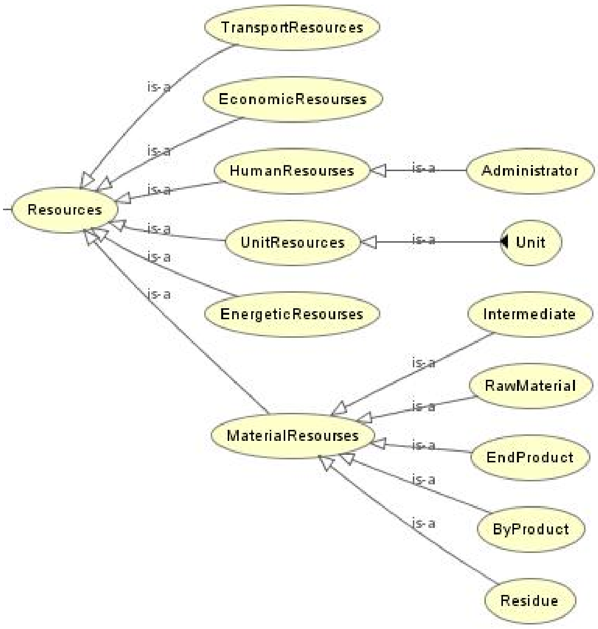


Figure 4.13: Resources taxonomy.

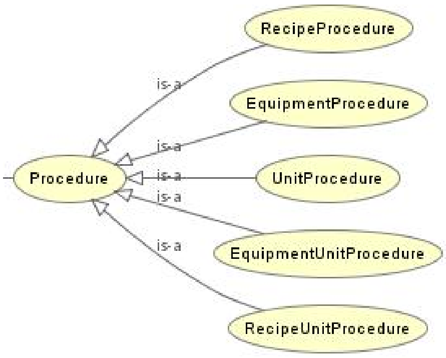


Figure 4.14: Procedure taxonomy.

- Table of verbs: In order to identify the properties that relate the classes of the enterprise ontology project, the following verbs have been proposed: to have, to be applied to, to carry, to derive in, to supply, to refer, to provide, to receive, to give, to send, to assign, to relate, to be part of, to demand.

- Table of properties: In order to identify the object type properties an interrelationship matrix has been constructed (see Appendix C). As a result of the properties identification process tables 4.3 and 4.4 contain the object type properties and the data type properties of the enterprise ontology project, respectively; as well as their domain and their range. Specifically a total of 140 object properties and 11 data properties are specified.

  [t]

Table 4.3: Object properties (domain & range).

Object Property	Domain	Range
derivesInSiteRecipe	GaneralRecipe;	SiteRecipe;
hasTransportationLink	Supplier; DistributionCenter; Market; Site;	TransportationLink;
demands	Market;	Demand;
hasRecipeUnitProcedure	RecipeProcedure;	RecipeUnitProcedure;
hasID_RecipeElement	RecipeElement;	IDRecipeElement;
hasProcedure	ProceduralElement;	Procedure;
hasArbitration	CoordinationControl;	Arbitration;
canCarryMaterialResource	TransportResources;	TransportedMaterials;
hasSupplier	RawMaterialCost;	Supplier;
hasUnitProcedure	Procedure;	UnitProcedure;
derivesInControlRecipe	MasterRecipe;	ControlRecipe;
referstoEquipmentModule	IDEquipmentModule;	EquipmentModule;
hasMaterialResource	LocationCapacity; InventoryMaterialResources;	MaterialResourses;
isAppliedAt	SupplyChainManagement;	Facility;
inverse_of_hasElementOf_27	Unit;	ProcessCell;
hasEquipmentModule	Unit; EquipmentModule;	EquipmentModule;
suppliesResources	Supplier;	Resources;
inverse_of_hasElementOf_29	EquipmentModule;	Unit; EquipmentModule;
referstohasID_Logic	IDLogic;	ProceduralLogic;
hasLocationCost	SCLocationManagement;	FacilityInvestmentCost;
hasEquipmentControl	BatchControl;	EquipmentControl;
hasProcessAction	ProcessOperation;	ProcessAction;
hasParameter	Formula; RecipeElement;	Parameter;
inverse_of_derivesIn_32	SiteRecipe;	GaneralRecipe;
inverse_of_derivesIn_33	MasterRecipe;	SiteRecipe;
inverse_of_derivesIn_34	ControlRecipe;	MasterRecipe;
hasProductionPolicy	Demand;	ProductionPolicy;
hasHumanResource	InventoryHumanResources;	HumanResourses;
hasEquipmentRequirement	MasterRecipe; RecipeElement;	EquipmentRequirement;
inverse_of_hasElementOf_14	ProcessManagement;	BatchProcess;
inverse_of_assingResourseTo	Unit; Batch;	Arbitration; Allocation;
hasProcessProduction	ProductionProcess;	BatchProcess; DiscretPartProcess; ContinuosProcess;
hasProcessControl	Unit; EquipmentEntity; ControlModule; EquipmentModule;	ProcessControl;
hasID_Equipment	Unit;	IDEquipment;
inverse_of_hasElementOf_6	EquipmentControl;	BatchControl;
inverse_of_hasElementOf_7	BatchControl;	ProcessControl;
hasRestriction	EquipmentRequirement;	Restriction;
inverse_of_hasElementOf_16	Process;	Facility;
referstoRecipeID	IDRecipe;	RecipeType;
inverse_of_hasElementOf_15	RecipeType;	BatchProcess;
inverse_of_hasElementOf_5	Batch;	BatchProcess;
hasBatch	BatchProcess;	Batch;
hasProcessManagement	BatchProcess;	ProcessManagement;

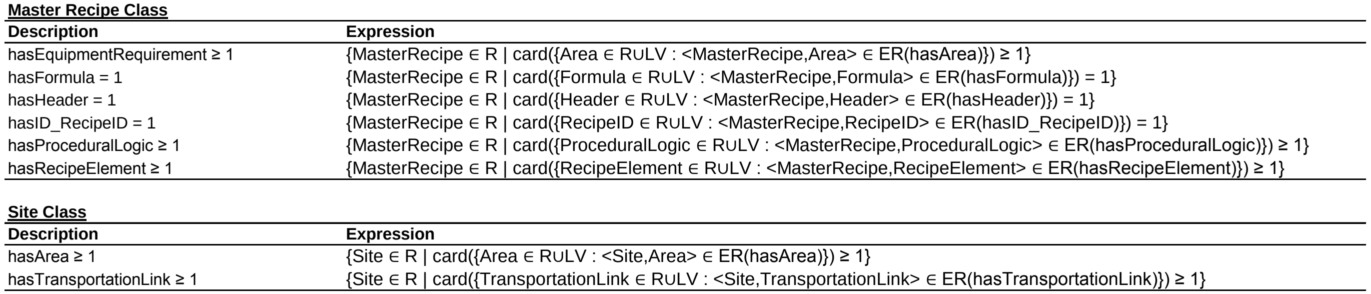
hasTransportationCost	TransportationLink;	InboundTransportationCost; InterfacilityTransportationCost; OutboundTransportationCost;
provideCost	RawMaterialCost;	RawMaterial;
hasID_EquipmentRequirement	EquipmentRequirement;	IDEquipment;
inverse_of_hasElementOf_1	EquipmentProcedure; EquipmentUnitProcedure; EquipmentOperation; ControlModule;	EquipmentControl;
hasDemand	IDMaterial;	Demand;
hasID_Logic	ProceduralLogic;	IDLogic;
hasProcessCell	Area;	ProcessCell;
hasRecipePhase	RecipeOperation;	RecipePhase;
hasSupplyChainManagement	Facility;	SupplyChainManagement;
hasRecipe	Unit; ProcessCell; EquipmentEntity; Site; ControlModule; EquipmentModule;	RecipeType;
isRuledBy	TransportResources;	TransportationLink;
hasLink_from	ProceduralLink;	IDProcessStage;
inverse_of_includesA_1	GaneralRecipe; ControlRecipe; MasterRecipe; SiteRecipe;	RecipeManagement;
hasIDFromRecipeElementType	RecipeElement;	IDRecipeElementType;
hasUnit	ProcessCell; InstalationCost;	Unit;
receiveInformationFrom IDRecipeElementType	IDRecipeElementType;	ProcessStage; RecipePhase; RecipeOperation;
hasTransport	TransportationLink;	TransportResources;
referstoRecipeElement	IDRecipeElement;	RecipeElement;
hasProcessOutput	BatchProcess;	Resources;
referstoEquipmentRequirement	IDEquipment;	EquipmentRequirement;
hasID_EquipmentModule	EquipmentModule;	IDEquipmentModule;
inverse_of_hasElementOf_30	ControlModule;	ControlModule; EquipmentModule;
hasModels	ProceduralElement;	Mode;
hasArea	Site;	Area;
hasBatchControl	ProcessControl;	BatchControl;
hasSite	Enterprise;	Site;
hasInventoryCost	SCInventoryManagement;	InventoryHoldingCost;
hasEqupmentOperation	EquipmentProcedure;	EquipmentOperation;
hasProductionOrder	Site;	ProductionOrder;
hasParameterSource	Parameter;	IDMaterial; ProcessParameter;
hasHeader	RecipeType; RecipeElement;	Header;
isAreaOf	Area;	Site;
sendMaterialTo	Supplier; DistributionCenter;	DistributionCenter; Market; Site;
receiveMaterialFrom	DistributionCenter;	Supplier; Site;
gives_control	ProcessControl;	Unit; EquipmentEntity; ControlModule; EquipmentModule;
hasResourceCost	Resources;	OtherAcquisitionCost; RawMaterialCost;
connectFacilities	TransportationLink;	GeographicPlacement;
refersTo	Unit;	EquipmentRequirement;
inverse_of_provideControlTo_5	BasicControl; CoordinationControl; ProceduralElement;	EquipmentControl;
hasEquipmentUnitProcedure	EquipmentProcedure;	EquipmentUnitProcedure;
assingResourceTo	Arbitration; Allocation;	Unit; Batch;
referstoEquipment	IDEquipment;	Unit;
hasInventory	SCInventoryManagement;	Inventory;
hasLink_to	ProceduralLink;	IDProcessStage;
hasProcess	Facility;	Process;
inverse_of_hasOrderedSetOf_15	RecipeOperation;	RecipeUnitProcedure;
inverse_of_hasOrderedSetOf_14	RecipeUnitProcedure;	RecipeProcedure;
inverse_of_hasOrderedSetOf_16	RecipePhase;	RecipeOperation;
inverse_of_hasOrderedSetOf_11	EquipmentUnitProcedure;	EquipmentProcedure;
hasProcessOperation	ProcessStage;	ProcessOperation;
referstoMaterial	IDMaterial;	Resources;

inverse_of_hasOrderedSetOf_10	Phase;	Operation;
inverse_of_hasOrderedSetOf_13	EquipmentPhase;	EquipmentOperation;
inverse_of_hasOrderedSetOf_12	EquipmentOperation;	EquipmentProcedure;
hasFormula	RecipeType;	Formula;
referedFrom	EquipmentRequirement;	Unit;
transportMaterial	TransportationLink;	IDMaterial;
hasRecipeOperation	RecipeUnitProcedure;	RecipeOperation;
hasPhases	Operation;	Phase;
receiveInformationFromIDLink	IDLink;	ProceduralLink;
provideID	ProceduralLink;	ID;
includesARecipes	RecipeManagement;	GaneralRecipe; ControlRecipe; MasterRecipe; SiteRecipe;
provideControl	EquipmentControl;	BasicControl; CoordinationControl; ProceduralElement;
hasEnergeticResource	InventoryEnergeticResources;	EnergeticResourses;
inverse_of_hasOrderedSetOf_9	Operation;	UnitProcedure;
inverse_of_hasOrderedSetOf_8	UnitProcedure;	Procedure;
hasTime	ProcessStage;	P_Time;
hasProceduralLogic	RecipeType; RecipeElement;	ProceduralLogic;
relatesToProduct	TransportedMaterials; Demand; ProductionOrder;	MaterialResourses;
hasEconomicResource	InventoryEconomicResources;	EconomicResourses;
hasID_Material	MaterialResourses;	IDMaterial;
inverse_of_hasOrderedSetOf_26	ProcessAction;	ProcessOperation;
inverse_of_hasOrderedSetOf_25	ProcessOperation;	ProcessStage;
isProductionProcessOf	Procedure;	Area;
inverse_of_hasOrderedSetOf_24	ProcessStage;	Process;
derivesInMasterRecipe	SiteRecipe;	MasterRecipe;
hasProductionans DistributionCost	SCProductionand DistributionManagement;	IndirectDistribution CenterCost; OtherAcquisitionCost; DirectManufacturingCost; IndirectManufacturingCost; DirectDistribution CenterCost; RawMaterialCost;
relatesToFacility	LocationCapacity; FacilityInvestmentCost;	Facility;
hasOperation	UnitProcedure;	Operation;
hasFacility	Enterprise; GeographicPlacement;	Facility;
hasProductionProcess	Area;	Procedure;
hasLocationParameter	SCLocationManagement;	LocationParameter;
hasControlModule	ControlModule; EquipmentModule;	ControlModule;
hasEquipmentParts	EquipmentControl;	EquipmentProcedure; EquipmentUnitProcedure; EquipmentOperation; ControlModule;
isProcessProduction	BatchProcess; DiscretPartProcess; ContinuosProcess;	ProductionProcess;
hasEquipmentPhase	EquipmentOperation;	EquipmentPhase;
isSiteOf	Site;	Enterprise;
hasAllocation	CoordinationControl;	Allocation;
hasID_ProcessStage	ProcessStage;	IDProcessStage;
hasBatchSize	Header;	BatchSize;
hasProcessInputParameter	ProcessStage;	P_Materials;
is_part_of_area	ProcessCell;	Area;
hasLink	ProceduralLogic;	ProceduralLink;
hasProcessOutputParameter	ProcessStage;	P_Materials;
hasID_RecipeID	RecipeType;	IDRecipe;
hasPhysicalProcessing	EquipmentEntity;	Unit; ProcessCell; ControlModule; EquipmentModule;
hasRecipeElement	RecipeType; RecipeElement;	RecipeElement;

Table 4.4: Data properties (domain & range).

DataProperty	Domain	Range
value	BatchSize; Inventory; Parameter; Restriction; Demand; LocationCapacity; SupplyChainCost; ProductionOrder;	float
link_type	ProceduralLink	string
max_value	BatchSize; ProcessParameter; Resources; LocationCapacity	float
min_value	BatchSize; ProcessParameter; Resources; LocationCapacity	float
salesPrice	Demand	float
existence	Facility	boolean
availability	Unit	boolean
priceUnits	Demand	string
due_date	Demand; ProductionOrder	data
unit_of_measure	BatchSize; PRocessOutput; ProcessParameter; Restriction; LocationCapacity; SupplyChainCost; ProductionOrder;	string
isDemanded	MaterialResources	boolean

- Table of restrictions (axioms): The restrictions build constraints between the properties above mentioned. Some examples of restrictions are quantifier restrictions, cardinality restrictions and hasValue restrictions. A total of 64 axioms have been defined for the proposed ontological model. For example, Figure 4.15 contains the main axioms of the Master Recipe and Area classes.



Master Recipe Class

Description	Expression
hasEquipmentRequirement ≥ 1	{MasterRecipe ∈ R | card({Area ∈ R∪LV : <MasterRecipe,Area> ∈ ER(hasArea)}) ≥ 1}
hasFormula = 1	{MasterRecipe ∈ R | card({Formula ∈ R∪LV : <MasterRecipe,Formula> ∈ ER(hasFormula)}) = 1}
hasHeader = 1	{MasterRecipe ∈ R | card({Header ∈ R∪LV : <MasterRecipe,Header> ∈ ER(hasHeader)}) = 1}
hasID_RecipeID = 1	{MasterRecipe ∈ R | card({RecipeID ∈ R∪LV : <MasterRecipe,RecipeID> ∈ ER(hasID_RecipeID)}) = 1}
hasProceduralLogic ≥ 1	{MasterRecipe ∈ R | card({ProceduralLogic ∈ R∪LV : <MasterRecipe,ProceduralLogic> ∈ ER(hasProceduralLogic)}) ≥ 1}
hasRecipeElement ≥ 1	{MasterRecipe ∈ R | card({RecipeElement ∈ R∪LV : <MasterRecipe,RecipeElement> ∈ ER(hasRecipeElement)}) ≥ 1}

Site Class

Description	Expression
hasArea ≥ 1	{Site ∈ R | card({Area ∈ R∪LV : <Site,Area> ∈ ER(hasArea)}) ≥ 1}
hasTransportationLink ≥ 1	{Site ∈ R | card({TransportationLink ∈ R∪LV : <Site,TransportationLink> ∈ ER(hasTransportationLink)}) ≥ 1}

Figure 4.15: Examples of axioms for the Master Recipe and Area classes.

- Tables of instances: Since one of the project objectives consist on the ontology usability, several case studies are presented in chapter 6. For each of them a table of instances is presented.

4.3.2 Formalization (ontology editor implementation)

The selected editor is Protégé (for Biomedical Informatics Research (BMIR), 2007), a description logic reasoning system as a tool for ontology editing and knowledge acquisition (Horridge et al., 2007). This system acts as an inference engine to check data consistency and validity (`http://protege.stanford.edu/`). The main reason for this choice is that Protégé is a widely used open-source ontology and knowledge base editor with a friendly user interface. Moreover, with this tool the representational and logical qualities that can be expressed in the built ontology allow the multiple inheritance concept and relation hierarchies; meta-classes; instances specification support; constraint axioms ala Prolog, F-Logic, OIL and general axiom language (PAL) via plug-ins. Furthermore the native or primary language used to encode the ontology is the OKBC model.

Furthermore, we used Collaborative Protégé, a Protégé extension that enables users who develop an ontology collaboratively to hold discussions, chat, annotate ontology components and changes, all as an integral part of the ontology development process. From Protégé, it is possible to export ontologies to other knowledge-representation systems, such as:

- RDF (Resource Description Framework) is mainly intended for use in the semantic web, but it has also been described as a content management technology, a knowledge management technology, a portal technology, and as one of the pillars of e-commerce (Klyne and Carroll, 2002).
- OIL (Ontology Inference Layer) is intended to solve the findability problem, support e-commerce, and enable knowledge management (Horrocks et al., 2000).
- DAML (The DARPA Agent Markup Language) focuses on supporting the semantic web, though one would assume that it also has other uses (Pagels, 2006).

Finally the creation of the ontology has been possible based on the Protègè's users guide (Horridge et al., 2007).

4.3.3 Integration

At this time, due to the lack of consensus among existing ontologies and no use of standards, it has not been possible the addition of other ontologies for their reuse. Besides, the models of the same domain vary depending on the construction method. For this reason, the inclusion of other ontologies' terminology has been disregarded until now.

4.3.4 Informatics implementation

The main objective established for developing the informatics system is its capacity to integrate different perspectives (e.g., different hierarchical decision levels) and the mappings between them. In this sense, the enterprise ontology project contemplates the enterprise system integration, where processes are categorized, the relationships between them are examined and imposed. Thus, the model instances must be retrieved from the decision maker optimization tools.

Technological architecture

The application development is based on the MVC (model view controller). The MVC is a standard software architecture that separates data from an application, the user interface, and control logic, into three distinct components called layers (Avgeriou and Zdun, 2005). The view, controller and model layers are separated clearly among them allowing the easy implementation of new features. Besides, this architecture allows the scaling of the infrastructure easily. Figure 4.16 shows the distribution of the layers and their corresponding servers.

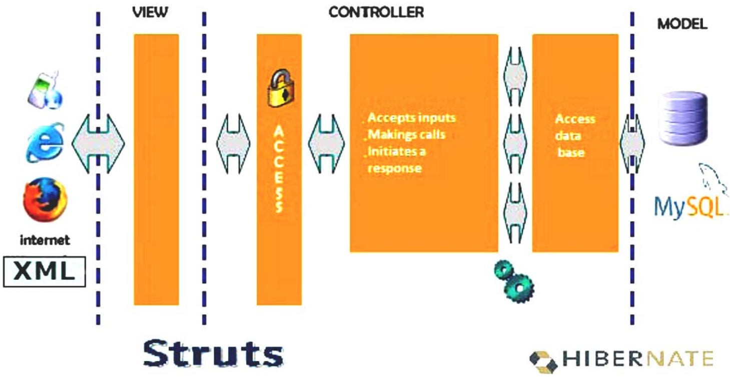


Figure 4.16: Model structure informatics

View layer The view layer manages the interaction with the final costumers and the delivery of information in different formats. This layer uses the Apache Struts framework, which is an open source for creating Java web applications (Gosling et al., 2005). The View represents the page design code, and the Controller represents the navigational code. The Struts framework is designed to help developers to create web applications that utilize the MVC architecture (Holmes, 2004).

Model layer Referred also as a business logic layer (BLL), also known as the domain layer, is a software engineering practice of compartmentalizing. The business logic layer is usually one of the tiers in a multi layer architecture. It separates the business logic from other modules, such as the data access layer and user interface. The programs that are running are located in the business layer. These programs receive user requests and send responses after the process has been executed. It is called business layer (and even business logic) because this is where all the rules that must be accomplished are set. This layer communicates with the view layer, to receive applications and present the results, and the data layer, to ask the database manager to store or retrieve data from it.

Controller layer (data & access) Data and access layer is where data reside and it is responsible for accessing it. It consists of one or more database administrators to perform all data storage, receiving requests for storing or retrieving information from the business layer.

For this layer the Hibernate framework has been used. Hibernate facilitates the storage and retrieval of Java domain objects via Object/Relational Mapping. Nowadays, Hibernate is a collection of related projects enabling developers to utilize POJO-style domain models in their applications in ways extending well beyond Object/Relational Mapping (Linwood et al., 2010).

As data base motor MySQL was chosen. The MySQL database server is the world's most widely used open source database. Its ingenious software architecture makes it extremely fast and easy to customize (Table, 2004). Extensive reuse of code within the software and a minimalistic approach to produce functionally rich features has resulted in a database management system unmatched in speed, compactness, stability and ease of deployment.

Application

For the applications, Java has been used as a high-level programming language. Java presents a good versatility, efficiency and security. Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems.

The application of the ontological model takes place inside the business layer. In this particular work the business layer is integrated by the planning and the controls tasks. Each task processes a XML recipe data. Once this recipe is processed by an external application, the recipe is sent to the next stage. The data could be saved in a data base and at the same time sent to the control level.

The applications of planning and control have an important module that uses the Case Base Reasoning (CBR) method, to collaborate in the decision task. At the same time this CBR helps to the information recovery acting as a cache or reusing the data information.

4.4 Study/Check phase

The five steps comprised in this phase according to section 3.2 are explained next.

4.4.1 Language conformity standard

Since the software for ontology construction and edition, Protègè, is based on the standards in accordance agreement with the rules from the world wide web consortium (W3C), this step of the check phase is automatically fulfilled.

4.4.2 Conceptually conformity standard

The enterprise ontology project is based on the ANSI/ISA standards, as well as widely recognized reviews in the PSE domain. For this reason, the conceptual conformity of the built project is guaranteed.

4.4.3 Reasoning

The support for debugging defects in OWL ontologies has been fairly weak. Common defects include inconsistent ontologies and unsatisfiable concepts. An unsatisfiable concept is one that cannot possibly have any instances or it represents the empty set (e.g., owl:Nothing). However, these errors can be detected automatically using a DL reasoner, which simply reports the errors, without explaining why the error occurs or how it can be resolved correctly.

The reasoning of the enterprise ontology project has been performed using RacerPro and several reasoners included in Protègè namely FaCT++, HermiT, Pellet and Pellet incremental. The reasoners performed the identification of the domain and range of properties (relationships) among the classes that compose the ontology. As a result, they detected problems such as:

- Inverse properties must have inverse domains and ranges.
- Missing disjoints on primitive subclasses.
- The transitivity of a property should also hold for its inverse.

In general, inconsistency was related to unsatisfiable class and properties description. Only if the taxonomic classification of the classes is checked and the ontology is consistency in accordance to the reasoners, the asserted individuals consistent could be checked as well.

Along the project development inconsistencies have been solved at the time they appeared. Having found defects in the ontology, their resolution has been carried out, requiring an exploration of remedies with a cost/benefit analysis. In this case, repair solutions that impact the ontology minimally have been generated. Particular care and effort was taken to ensure that ontology repair is carried out efficiently. Table 4.5 contains the reported times by the reasoners for checking consistency once the ontology has been debugged.

Table 4.5: Reasoners time for consistency checking.

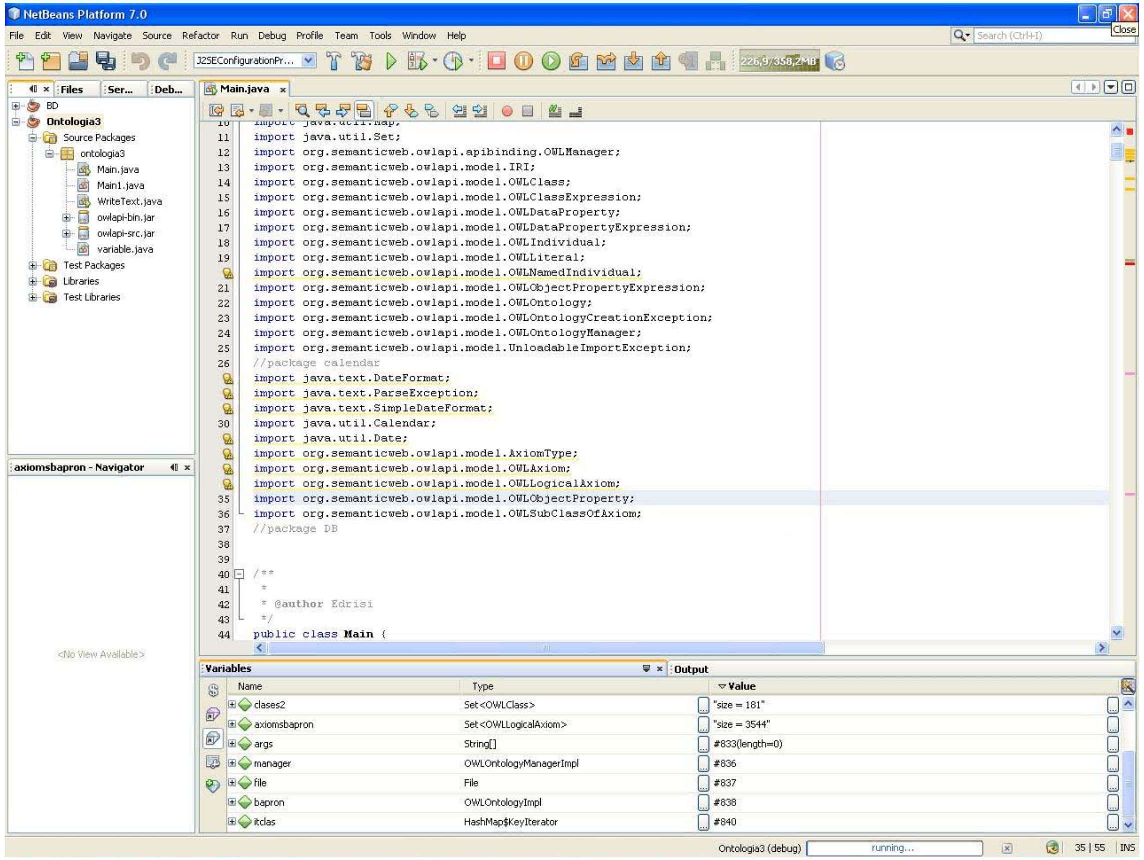
Reasoner name	time[sCPU]
Pellet	1.938
FaCT++	0.204
HermiT	16.266
Pellet (Incremental)	3.828

4.4.4 Performance (application oriented)

In order to build the informatics application, the software NetBeans platform has been used for editing the java code necessary to program the functions for the particular task that the enterprise ontology project was designed for (Figure 4.17).

Thus the OWL API interface (Horridge et al., 2007) has been adopted as a base code for its OWL APIs functionality. The OWL API is a Java interface and implementation for the W3C Web Ontology Language OWL. The latest version of the API is focused towards OWL 2 which encompasses, OWL-Lite, OWL-DL and some elements of OWL-Full. The OWL API is open source. The interface allows to access to the ontology within a java environment.

The complete application code is given in (Appendix D). The results of running the informatics application may be saved in a Data Base for facilitating the access to them. These results are used as inputs of the optimization tools for the decision makers. Thus the results of the optimization are available to update and be included in the data instantiation of problem in the ontological framework.



```
import java.util.Map;
import java.util.Set;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLClassExpression;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLDataPropertyExpression;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLLiteral;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectPropertyExpression;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.UnloadableImportException;
//package calendar
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import org.semanticweb.owlapi.model.AxiomType;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLLogicalAxiom;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLSubClassOfAxiom;
//package DB


/**
 *
 * @author Edrisi
 */
public class Main {
```

Name	Type	Value
clases2	Set<OWLClass>	"size = 181"
axiomsbapron	Set<OWLLogicalAxiom>	"size = 3544"
args	String[]	#833(length=0)
manager	OWLOntologyManagerImpl	#836
file	File	#837
bapron	OWLOntologyImpl	#838
itclas	HashMap$KeyIterator	#840

Figure 4.17: Screen shot of NetBeans environment for the informatics application.

4.4.5 Revise

The results of the optimization were analyzed as an issue of validation and comparison of behavior. The results refer to different case studies which belong to the domain defined in the enterprise ontology project.

The use of the enterprise ontology project enables the integration of the different decision levels and supplied the process state information to different levels in the decision-making hierarchical structure, which leads to the following improvements:

- Behavior: The efficiency of the reactive policies are improved by updating the decision level's state in a most effective way, which leads to an increase in process yield.
- Language: The information presented proper, common and standardized language within the hierarchical levels, regardless of its origins.
- Documentation: Ontology provides a certain and easy description for formal (read by a computer) and informal (presented in a human-readable form) specifications of the system elements' content.
- Storage: Efficient access is provided to process databases using a database management system and to establish the links and locations of these databases.
- Navigation: The Web-Protégé portal provides the option of interaction among potential users of the ontology, by interchanging some notes about the different parts.
- Data: Significant data can be mined from the whole amount of available data that flows along the hierarchical structure levels of the enterprise. Thus it is possible to use it as information quality.

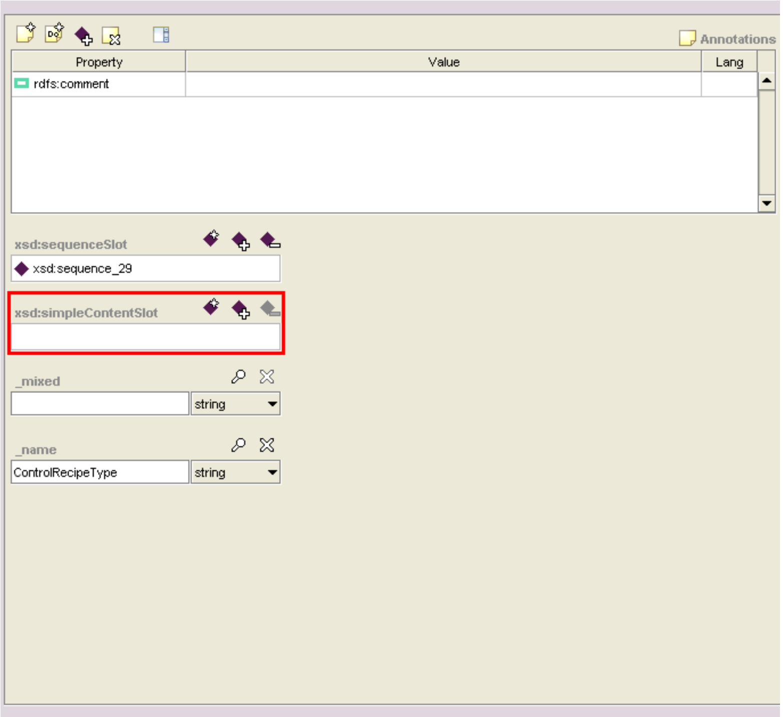
A comparison between the process of how different levels interact in a traditional way and a proposed one has been done. An average improvement of 20% in the number of steps for carrying out each process has been obtained, leading to savings in process time ranging between 70-80%.

4.5 Act phase

4.5.1 Implementation

- Performance (application environment): After the creation of the enterprise ontology project, different applications have been performed in benchmark problems of the literature comprising the supply chain, planning & scheduling and process control domains. the behavior of the framework is thoroughly reported in chapter 6.

  A remarkable aspect consists of the fact that when a system or a process is instantiated within the ontology structure during the problem implementation, the relationships defined by axioms and rules ensure that every part or instance follows the standards requirements. This can be defined as an automated way to standardize a particular process. Even more, those parts that are compulsory within the model are demanded during the problem instantiation. Specifically, once a class is instantiated, those other required classes are automatically demanded by the ontology. Such feature is achieved by creating the adequate axioms. For example, as shown in Figure 4.18, the slot information is required and requested by the ontological model when a control recipe instantiation is performed.



xsd:sequenceSlot: xsd:sequence_29

_mixed: string

_name: ControlRecipeType string

Figure 4.18: Protege required parts in order to instance the process.

As a conclusion of this step, it can be stated that the enterprise ontology project is general enough to be used and reused in any enterprise wide optimization case.

- Distribution of the new ontology: A users manual contains in general terms the procedures for:
  - Instantiating a real problem.
  - Checking consistency.
  - Generating input files for the decision maker. This file can be used for machines (formal specification) or people(informal specification).

Appendix E contains the users manual for the enterprise ontology project.

4.5.2 Maintenance

The maintenance phase is part of the continued use of this project. Hence, the formalization and documentation of possible changes are updated when improvements in the domain application are made.

4.6 Re-planning phase

The results of this phase are included in the previous subsections, which present the final structure of the pursuit project. However the project has been re-planned several times adding new functionalities to the studied domain. The main stages are:

1. Construction oriented to the control and scheduling levels.
2. Addition of the planning level (site recipe issues).
3. Addition of the supply chain level (strategic issues).

Chapter 5

Decision support system

As explained Section 2.4 the traditional components of decision support systems include: (i) data management capabilities, (ii) modeling functions, and (iii) interface designs (Shim et al., 2002). Precisely, by means of the systematic solutions, a decision support systems tool capable of handling much softer information and broader concern than the mathematical models and knowledge systems handled until now can be developed.

Model management systems and knowledge-based decision support systems have use techniques as artificial intelligence and expert systems to provide smarter support for the decision-maker. Furthermore, the Internet, the Web and telecommunications technology can be expected to result in organization environments that will be increasingly more global and complex.

In this thesis, a model is created toward closing the gap between transactional and analytical models which are used in the technical and organizational parts. It also integrates the different hierarchical levels within the enterprise structure, making information quality available. As a result, multiple and varied decision perspectives may be available within the decision making task, increasing the speed response of the decision support system.

This chapter aims to explain how decision support systems may be involved in the exploitation of the capabilities of the proposed ontological model. Thus, the specific decisions that are to be assisted at each decision level of the hierarchical structure are thoroughly defined.

5.1 Process control decisions

This thesis handles process control decisions related to procedural and coordination control. Therefore, the implementation of the sequence of control steps in the equipment modules, as well as the transition between control recipes are dealt. Such decisions are specified in the control recipes, which receive data from the scheduling level, and provide data with the actual plant state. Next, the information flow and procedures involved in this decision level along with their relationship with other decision levels are described.

The data requirement for this decision level as well as the main classes that must be instantiated for the coordination control are given in table 5.1.

Table 5.1: Information provided by the ontology to the control level.

Concept	**Ontology implementation (classes involved)**
Capacity	P_Capacity
Pressure	P_Pressure
Temperature	P_Temperature
Transfere rate	P_TransfereRate
Valve position	P_ValvePosition
Mixing speed	P_MixingSpeed
Power	P_Power
Coordination cntrol	CoordinationCntrol
Equipment control	EquipmentControl
Procedural control	ProceduralControl
Procedural element	ProceduralElement
Operation	Operation
Phases	Phases
Equipment operation	EquipmentOperation
Recipe operation	RecipeOperation
Equipment phase	EquipmentPhase
Recipe phase	RecipePhase

The decisions at this level are concerned with the implementation of the control recipe according to the results from the scheduling decision level, and the update of the current plant state. The scheme of the control recipe is written using Matlab (Mathworks, 2009) in XML language. Therefore, the coordination control manages the batch operation by using the control recipe scheme and using the data contained in the ontological model. The data are managed by Matlab, which provides a set of information that can be linked and processed by the Java interface, and also related to the appropriate classes and properties in the ontology.

Thus, the results from the scheduling function are on their turn passed from the scheduling function to the ontology by the Java environment providing the control function with the necessary data. As a whole, it is an iterative procedure for decision making, which involves the control and the scheduling levels (Figure 5.1). The plant workout as well as the equipment model are executed in Matlab/Simulink environment.

Case studies 1 and 2 in chapter 6 detail the flow of information during the performance of this decision level.

5.2 Planning & scheduling decisions

The scheduling decisions that are handled in this thesis work are concerned with the sequencing and timing of multi-product ans multi-purpose batch plants. It is also deals with the scheduling task and rescheduling after an unexpected event occurs.

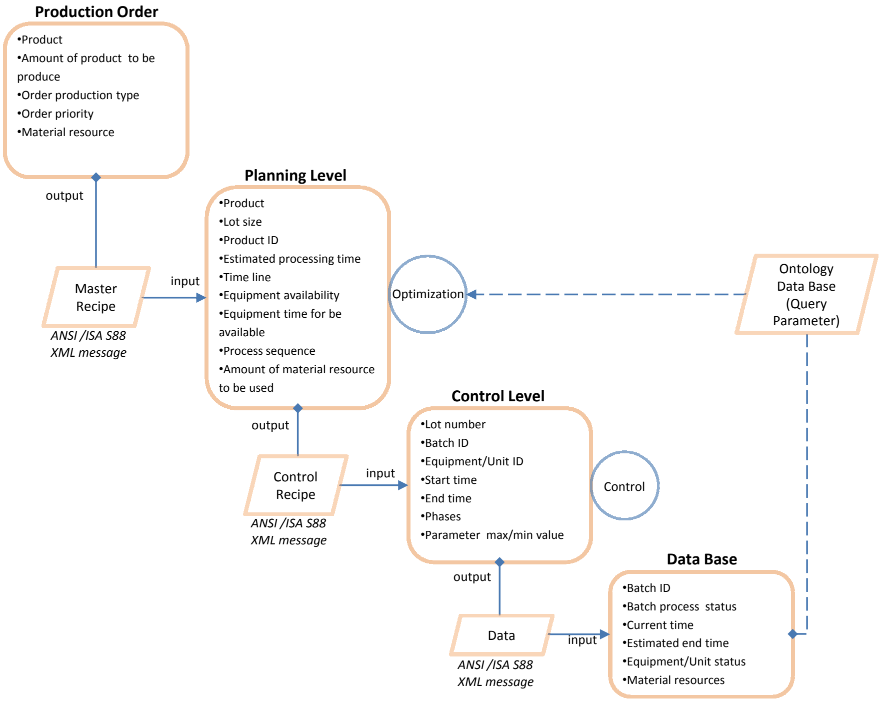


Figure 5.1: Information required by the recipe at different levels.

A large number of approaches has been presented in the literature to model and solve planning and scheduling problems. On the one hand, mathematical representations, graphic or artificial intelligence based models can be used. On the other hand, mathematical programming, logic-based methods, heuristics, meta-heuristics, or hybrid methods have been applied for solving the planning and scheduling problems (Mendez et al., 2006).

In this thesis, an optimization based decision support system using mathematical programming is adopted for solving the scheduling problem. In the literature, there are several formulations such as the general precedence model, the continuous and discrete RTN formulation, or unit-specific time event. Specifically, in this thesis, the continuous time STN-formulation presented by Maravelias and Grossmann (2003) has been considered, since it is able to represent multi-product and multipurpose plant configurations. However, any other formulation or optimization approach, able to represent the problems' features, could have been used to solve the scheduling problem.

The continuous time STN representation is based on the definition of a common time grid that is variable and valid for all shared resources. To guarantee the feasibility of the material balances at any time during the time horizon of interest, the model imposes that all tasks starting at a time point must occur at the same time T_n, where n is the set of time points occurring at unknown time T_n, $n = 1, 2, ..., N$. The ending time does not necessarily have to coincide with the occurrence of a time point n, except for those tasks that need to transfer the material with a zero wait time policy. The model adopts two binary variables to denote at which time point a given task i starts and finishes. Moreover, it is necessary to define the batch size of a given task at the beginning, finishing and during the processing time of the task. Thus, the the quantity of resources available at each time point can be monitored.

In this thesis, the ontological model has been used to represent the scheduling problem and adequately translated using the Java environment to be exploited by the mathematical programming tool. Indeed, the formulation has been implemented in GAMS and solved using the MILP solver CPLEX 9.0, since the posed problems (their specific formulations) are lineal.

5.2.1 Data requirement

The data required for taking decisions at the scheduling level are specified next. The steps comprised in the usability to scheduling given in this ontological model are common to all cases. Specifically, once the problem is instantiated, the scheduling function requires the following information:

- Capacity: It refers to the raw materials quantities available in the production plant, and the maximum storage capacity of the intermediates, residues and final products.
- Demand: It contains the quantity of each product required by the customers.
- Due-date: It contains the date that each final product has to be delivered to the customer.
- Product-stage-unit: It contains the set of units that is available at each stage of the production process for all products.
- Quantities in/out: It refers to the quantities of the raw materials, intermediates and final products that are necessary to perform/ that are delivered at each stage of the production recipe of each product according to the material balances.
- Processing time: The time necessary to perform the tasks of a given stage.
- Stage-process: It contains the set of stages that are contained in each master recipe of the products.
- Time horizon: It refers to the time horizon for the scheduling task.
- Unit availability. It contains the time at which the unit is available.

Therefore, the Java code is programmed for generating the input files for the scheduling optimization tools. Table 5.2 contains the sources from the ontological model that are used to provide the optimization framework with the necessary data. Finally, the optimization software can be executed. On the whole, any scheduling problem can be modeled by the ontological framework, and next solved by the decision maker tools.

In Case studies 2 to 5 the usability of the ontological model is demonstrated and the advantages obtained are highlighted.

Table 5.2: Relationship between optimization inputs and classes of the ontology.

Concept	**Set**	**Ontology implementation (classes involved)**
Recipe	p	∀ MasterRecipe → RecipeID
Stages	s	∀ RecipeElement |RecipeElementType = ProcessStage → RecipeElementID
Units	u	∀ EquipmentRequirement ∈ GeneralInformation → EquipmentRequirementID
Materials	i	∀ RawMaterial ∈ ProcessInputs → MaterialsID ∀ Intermediate ∈ ProcessInputs → MaterialsID ∀ By_Product ∈ ProcessOutputs → MaterialsID ∀ End_Product ∈ ProcessOutputs → MaterialsID ∀ Residue ∈ ProcessOutputs → MaterialsID
Stage-process	p,s	∀ ProcessStage → ProcessStageID
Raw materials	i	∀ RecipeElemnt ∈ MasterRecipeID → RecipeElementID ∀ RawMaterial ∈ ProcessInput → RawMaterialID
Final products	i	∀ FinalProduct ∈ ProcessOutput → MaterialID
Product-Stage-Unit	p,s,u	∀ EquipmentRequirement ∈ ∀ RecipeElement ∈ MasterRecipe → EquipmentRequirementID
Process-stage-input	p,s,i	∀ ProcessStage ∈ RecipeElement ∈ MasterRecipe → ProcessInputParameter = ParameterSource → ID
Process-stage-output	p,s,i	∀ ProcessStage ∈ RecipeElement ∈ MasterRecipe → ProcessOutputParameter = ParameterSource → ID
Quantity- process-stage-input	p,s,i	∀ Material ∈ ProcessStage ∈ RecipeElement ∈ MasterRecipe → ProcessInputParameter = ParameterSource → Value
Quantity- process-stage-output	p,s,i	∀ Material ∈ ProcessStage ∈ RecipeElement ∈ MasterRecipe → ProcessOutputParameter = ParameterSource → Value
Processing time	p,s,u	DataBasefor $\forall u; \forall s: \forall p$
Storage capacity	i	DataBasefor ∀ Material
Demand	i	DataBasefor ∀ FinalProduct(i)

5.3 Supply chain decisions

In this thesis, the decisions involved in the strategic level are related to the number of facilities to be opened, the increase of their capacity at each time period, the linkages among them, the assignment of manufacturing and distribution tasks to the networks nodes, and the amount of final products to be sold, among others. Such decision environment can be friendly captured by the ontological environment.

In general, strategic decisions for determining the optimal SC network structure are key for the later optimization of SC operations. Traditional approaches available in literature addressing this problem usually utilize as departing point a rigid pre-defined network structure which may restrict the opportunities of adding business value. In this thesis, a flexible formulation approach which translates a recipe representation to the SC environment is used for the design and retrofit of SC (Lainez et al., 2009). Such design-planning model is analytical optimization model, based on the STN concept for representing SC networks, containing equations related to the mass balances, the design, the capacity and the markets and suppliers, as well as a complete set of economic performance metrics grouped in operating revenue, operating cost, and capital investment.

Such analytical optimization model must be provided with the necessary information regarding the SC structure, which is derived from the ontological model and the related data contained in the database. Additionally, the ontological model optimizes the way in which the databases are distributed along the enterprise structure. As a result, databases are well located and their data are easily available and can be transformed into valuable information.

The optimization model has been implemented in GAMS and solved using CPLEX 9.0. The Java application is used for generating the required inputs of the model, specifically the code generates the .txt files which are called by the optimization problem (Lainez et al., 2009).

The usability of the ontology for SC decisions is illustrated in case study 6 of chapter 6.

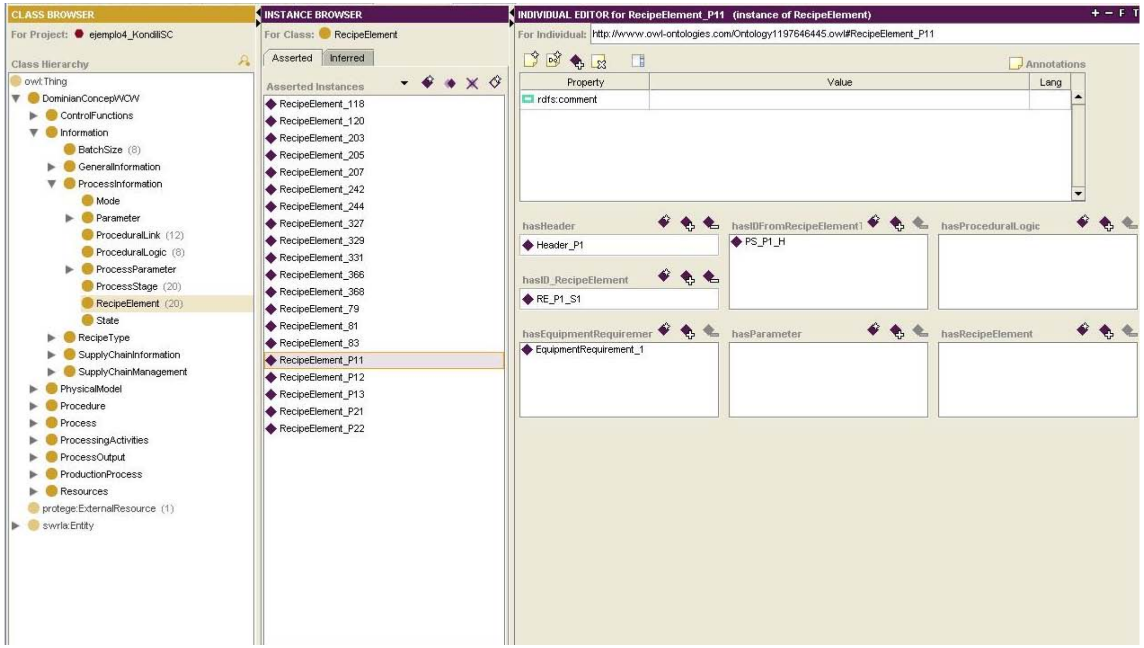
5.3.1 Data requirement

The specific information needed at this decision level is presented in Table 5.3.

The "production task" model element is part of the ontological model as shown in Figure 5.2, specifically the recipe element. It is necessary to export such instances to a format readable by the analytical system, namely a .set file (Table 5.4). Therefore, the adequate Java code (Figure 5.3) must be written in order to create the necessary input files.

Table 5.3: Information provided by the ontology to the analytical model.

SC model concept	**Ontology implementation (classes involved)**
States	RawMaterial, Intermediate and EndProduct
Locations	Site and DistributionCenter
Facilities	Supplier, SitePlacement and Market
Markets	Markerts
Activities	RecipeElements and TransportedMaterials
Technologies	Units and TrasportResources
Equipment	Units
Final products	EndProduct
Raw materials	RawMaterial
Distribution tasks	TrasportedMaterials
Production tasks	RecipeElements
Supplier sites	Supplier
Production sites	Sites
Distribution centers	DistributionCenter
SC model concept	Ontology class
Capacity transports	TransportedMaterials(relatesToProduct)
Cost raw material	RawMaterialCost
Facility investment cost	FacilityInvestmentCost (relatesToFacility value)
Facility location relationship	SitePlacement(hasFacility)
Market location relationship	MarketPlacement(hasFacility)
Market price	Market(Demands sales price)
Max capacity technology	TrasportResources(canCarryMaterialResource max value)
Min capacity technology	TrasportResources(canCarryMaterialResource min value)
Process inputs	RecipeElement(hasProcessInputParameter)
Process outputs	RecipeElement(hasProcessOutputParameter)
SC demand	Market(Demands value)
Supplier capacity	LocationCapacity(hasMaterialResource value)
Transport costs	TransportationLink (hasTrasportationCost)
Transport resources	TransportResources(canCarryMaterialResource)
Max facility capacity	LocationCapacity(hasMaterialResource max value)
Min facility capacity	LocationCapacity(hasMaterialResource min value)



For Project: ejemplo4_KondiliSC

Class Hierarchy

```
owl:Thing
▼ DominianConcepWCW
    ▶ ControlFunctions
    ▼ Information
        BatchSize (8)
        ▶ GeneralInformation
        ▼ ProcessInformation
            Mode
            ▶ Parameter
            ProceduralLink (12)
            ProceduralLogic (8)
            ▶ ProcessParameter
            ProcessStage (20)
            RecipeElement (20)
            State
        ▶ RecipeType
        ▶ SupplyChainInformation
        ▶ SupplyChainManagement
    ▶ PhysicalModel
    ▶ Procedure
    ▶ Process
    ▶ ProcessingActivities
    ▶ ProcessOutput
    ▶ ProductionProcess
    ▶ Resources
  protege:ExternalResource (1)
▶ swrla:Entity
```

For Class: RecipeElement

```
RecipeElement_118
RecipeElement_120
RecipeElement_203
RecipeElement_205
RecipeElement_207
RecipeElement_242
RecipeElement_244
RecipeElement_327
RecipeElement_329
RecipeElement_331
RecipeElement_366
RecipeElement_368
RecipeElement_79
RecipeElement_81
RecipeElement_83
RecipeElement_P11
RecipeElement_P12
RecipeElement_P13
RecipeElement_P21
RecipeElement_P22
```

For Individual: http://www.owl-ontologies.com/Ontology1197646445.owl#RecipeElement_P11

Property	Value	Lang
rdfs:comment		

hasHeader: Header_P1

hasID_RecipeElement: RE_P1_S1

hasIDFromRecipeElement1: PS_P1_H

hasProceduralLogic:

hasEquipmentRequiremer: EquipmentRequirement_1

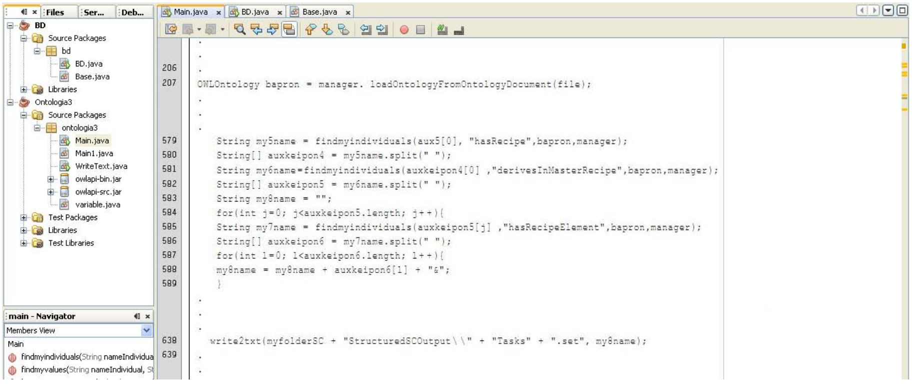
hasParameter:

hasRecipeElement:

Figure 5.2: View of the Protègè interface containing the instances required for defining the model element "production tasks".

Table 5.4: Elements of the model element "tasks".

tasks.set
RecipeElementP11
RecipeElementP12
RecipeElementP13
RecipeElementP21
RecipeElementP22



```
.
.
.
OWLOntology bapron = manager. loadOntologyFromOntologyDocument(file);
.
.
.
   String my5name = findmyindividuals(aux5[0], "hasRecipe",bapron,manager);
   String[] auxkeipon4 = my5name.split(" ");
   String my6name=findmyindividuals(auxkeipon4[0] ,"derivesInMasterRecipe",bapron,manager);
   String[] auxkeipon5 = my6name.split(" ");
   String my8name = "";
   for(int j=0; j<auxkeipon5.length; j++){
   String my7name = findmyindividuals(auxkeipon5[j] ,"hasRecipeElement",bapron,manager);
   String[] auxkeipon6 = my7name.split(" ");
   for(int l=0; l<auxkeipon6.length; l++){
   my8name = my8name + auxkeipon6[l] + "&";
    }
.
.
.
  write2txt(myfolderSC + "StructuredSCOutput\\" + "Tasks" + ".set", my8name);
.
.
```

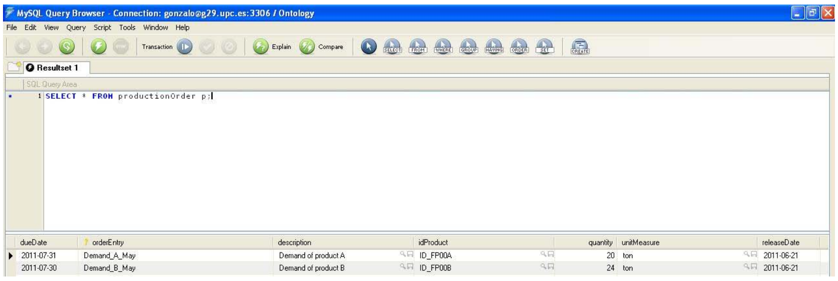
Figure 5.3: Example of the Java code developed to give the model element "tasks" to the analytical model.

5.4 Data and information management

Interoperability among different decision support tools is a critical aspect in the daily operation of enterprises. Thus, data bases are used to store the values related to specific and relevant aspects of the enterprise environment. In order to enhance the management of data by the ontological framework, it is important to consider the link and communication among data bases and the ontology. As a result, the decision support tool will benefit from a higher data availability and their subsequent interpretation as information quality.

In this thesis, the data which are introduced in the decision support systems are directly problem instances of the ontological model, whose dynamic values (those which are frequently updated) are read from different databases. What is more, an automatic order of the net of databases, which many times are spread along the different hierarchical decision levels, is achieved since every database is adequately related to the corresponding part of the ontological model. Every relationship between the dynamic value, e.g. demand data property in the ontology, and its corresponding numeric value stored in the data base is easily programmed in Java language (Figure 5.4).

An additional feature of the data management framework consists of the possibility of introducing the results of the decision support systems in the databases for its further exploitation by other decision support tool. For example, the results of the supply chain decisions are stored in the corresponding databases of the plants, and



```
SELECT * FROM productionOrder p;
```

dueDate	orderEntry	description	idProduct	quantity	unitMeasure	releaseDate
2011-07-31	Demand_A_May	Demand of product A	ID_FP00A	20	ton	2011-06-21
2011-07-30	Demand_B_May	Demand of product B	ID_FP00B	24	ton	2011-06-21

Figure 5.4: Screenshot of the MySQL database interface.

next the data for building the model in the scheduling decision support tool are based on the ontological model and the data stored in the databases (Figure 5.5).

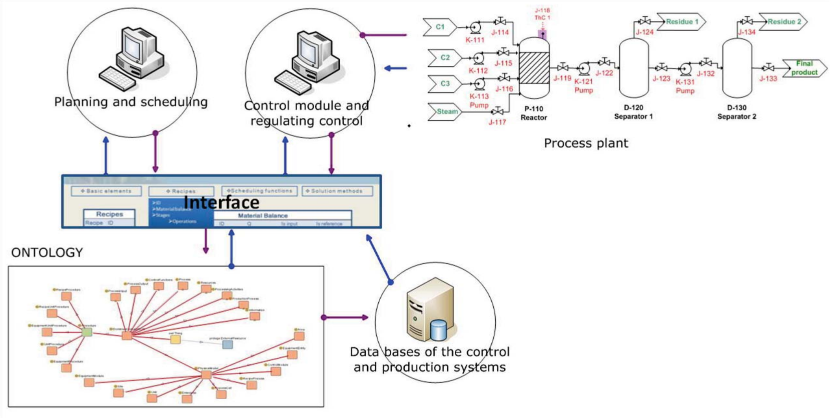


Figure 5.5: Scheme of the relationships among the different actors of the ontological framework and the database.

An important advantage regarding database consistency with the ontology is the design of the database tables and field structures based on the terminology and knowledge presented in the ontology. In this way, a better database net structure would be achieved.

A proposed additional decision support tool which can be directly exploited by the ontological framework and the database is the case based reasoning. Such tool is an artificial intelligence technique based on the fact that similar problems have similar solutions. Therefore, problem-solution pairs are recorded in the form of "cases" which are recorded in the case base repository.

Therefore, the importance of an adequate storage of the optimization results is crucial. The CBR employed in this thesis is the FreeCBR package (Johanson, 2011), which contains an API able to be used in the Java environment. The case based reasoner is applied in Case study 2 of Chapter 6.

5.5 Remarks

The defined decision support system shows several drawbacks and advantages when applied at the different decision levels.

On the one hand the main inconvenient of the proposed framework consist of the additional modeling time, since a high level of detail is required for modeling the problem. In addition it is necessary to be familiar with the ontological framework in order to know what information to extract and where it is located.

On the other hand a large number of advantages can be mentioned derived from the application of the ontological framework to the decision support tools. Since the model is more complete it is easier to integrate the different decision levels, as well as to consider other data. Thus, the change of the parameters is straight forward. THe framework also facilitates the establishment of relationships and properties between the data bases and the models. Moreover, the application of any model or any kind of model at any decision level can be generalized. Even more, the change in the physical structure of the problem (adding complexity) is also easy to carry out. Finally, the standardization of concepts and vocabulary allows the generalization in the application of the decision support tools regardless of the specific process structure by the consensus in the vocabulary.

Chapter 6

Case studies

6.1 Introduction

In this chapter, the application of the ontological framework is demonstrated in several process, plant and enterprise configurations related to different case studies using the decision support tools described in Chapter 5. The ontology has been created from scratch, so it has evolved along the different iterations of the re-planning phase, as explained in Chapter 4. This can be observed in the different models through the time line and their corresponding case studies, which are presented in order of increasing completeness of the ontological model.

Thus, the contributions of the proposed integrated information environment to the supply chain, scheduling and process control decision-making levels are shown, emphasizing the following features: re-usability, usability and more efficient communication.

The re-usability consists of the instantiation of different problems using the same modeling framework. The specific information regarding each case study is developed in every instantiation description.

The usability refers to the capacity of transforming the data contained in the problem instantiated in the model into valuable information (information quality). Precisely, the ontology allows the representation in a single model of the whole enterprise domain. The active parts of the model are determined by the final goals and by the use of the ontology in the enterprise level at which it will be applied.

Furthermore, the development of an ontological representation for the enterprise domain leads to the centralization of information and the coordination of the different hierarchical decision levels, and it may improve the time for reaching the desired data from the enterprise. Such features are illustrated in the explanation of the presented case studies.

An additional advantage of the use of an ontology in the enterprise domain consists of the possibility to work in a web environment, which is emerging as a very important decision support system platform. As a result, technological barriers can be reduced and decision support systems may be available to managers and staff users which are geographically distributed at a relatively lower cost. Even more, enterprise-wide decision support systems can be implemented in geographically dispersed companies for improving their overall results.

Moreover, the ontological framework stands for a bridge in natural language between the reality representation and any informatics framework. Thus, the information is available in multiple informatics formats. This means the translation of knowledge and know-how to an efficient way to manage domain information flow.

6.2 Model 1: Information integration

This model contains the elements described in the ANSI/ISA-88 specifically the physical, procedural, process and recipe model (found in Appendix B) and their corresponding relationships (found in Appendix C). As a result, it consists of 92 classes, 10 axioms, and 132 properties. This ontology is the result of the first completion of the proposed PDSA based methodology described in chapter 4, before the re-planning phase. In this approach the management of recipes is performed by external files written in xml code. As a main feature obtained in this first ontology draft, the integration of the control and the scheduling hierarchical decision levels has been achieved. This means that the different data found within those two decision levels are shared and carried to the right places that require a specific data.

This model has been applied to two different process plants for checking the usability of the ontology. Thus, the capabilities of the ontology for acting as a connector (model integrator) agent between the scheduling and the control levels data bases have been explored.

6.2.1 Case Study 1: PROCEL scheduling

A batch pilot plant (PROCEL), which is a basic environment for open simulation and optimization in a real time environment package scenario, is located at the laboratory facilities at the UPC Chemical Engineering Department. PROCEL provides an appropriate scenario for the first evaluation of the ontology performance from a qualitative and usable perspectives and for studying and developing new process strategies. Indeed, the plant has been adequately instantiated proving the ontology principle of usability.

This case study deals with the production of three chemical products, each with different production requirements (Table 6.1). The production system comprises 3 principal processing units, namely two reactors and one tank. The detailed piping and instrumentation diagram of the process is shown in Figure 6.1. It is considered one recipe for each product. Nominal processing times, transfer times (Table 6.2) and cleaning times (Table 6.3) are considered fixed for initial scheduling. However, they will be subject to changes, if required, to react to unexpected events. The changes follow the corresponding models by which the process is updated according to the information received from the plant. In Table 6.4 some control set points are taken into account, which are the maximum values for maintaining the plant under control.

The data contained in the ontological model can be mined by any external informatics agent. Such retrieval of information by an informatics systems is referred as the formal exploitation of the ontological model. Specifically, the data contained in the ontology can be used for the creation of files for further use, for example the

Table 6.1: Batches processing times [min].

Product	i1		i2		i3	
Stage	Unit	Time	Unit	Time	Unit	Time
l1	j1	30	j2	24	j3	50
l2	j2	60	j3	12	j2	12
l3	j3	59	j1	54	-	-

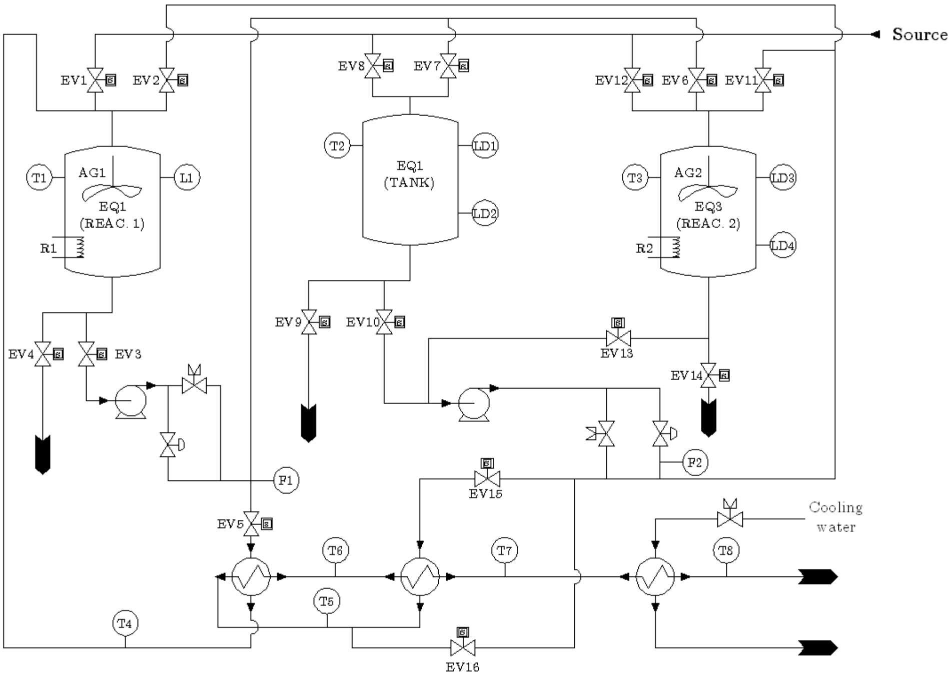


Figure 6.1: Piping and instrumentation diagram of the flow shop plant of the PROCEL plant.

Table 6.2: Pipes transfer times [min].

From-to	Unit (j1)	Unit (j2)	Unit (j3)
Unit (j1)	0	5.8	9.6
Unit (j2)	10.4	0	5.6
Unit (j3)	10.3	11.8	0

temperature parameter may be required by the master recipe and the control recipes. Therefore, Figure 6.2 shows how the information is shared both by the master recipe (unique file), and the control recipe. Such exploitation is achieved by the application of a single and standardized model at these two decision levels (scheduling and control).

Table 6.3: Cleaning time [min].

Product	Unit (j)	Time
i1	j1	20
i2	j2	10
i3	j3	20
i1	j1	40
i2	j2	20
i3	j3	40
i1	j1	10
i2	j2	5
i3	j3	10

Table 6.4: Control set points.

Control Parameter	**Value**
Descharge EQ1 LT1min	2.4 dm^3
Charge QE1 LT1max	4.4 dm^3
Hold tank time	60 s
SP clean EQ1 time	90 s
SP clean EQ3 time	120 s
Heat EQ1 SP T1	50ºC
Heat EQ3 SP T3	50ºC
Heat EQ1 SP R1	100%
Heat EQ3 SP R3	50%
Heat EQ1 SP AG1	100%
Heat EQ3 SP AG2	20%

Control Set Points (TOP: operation time)

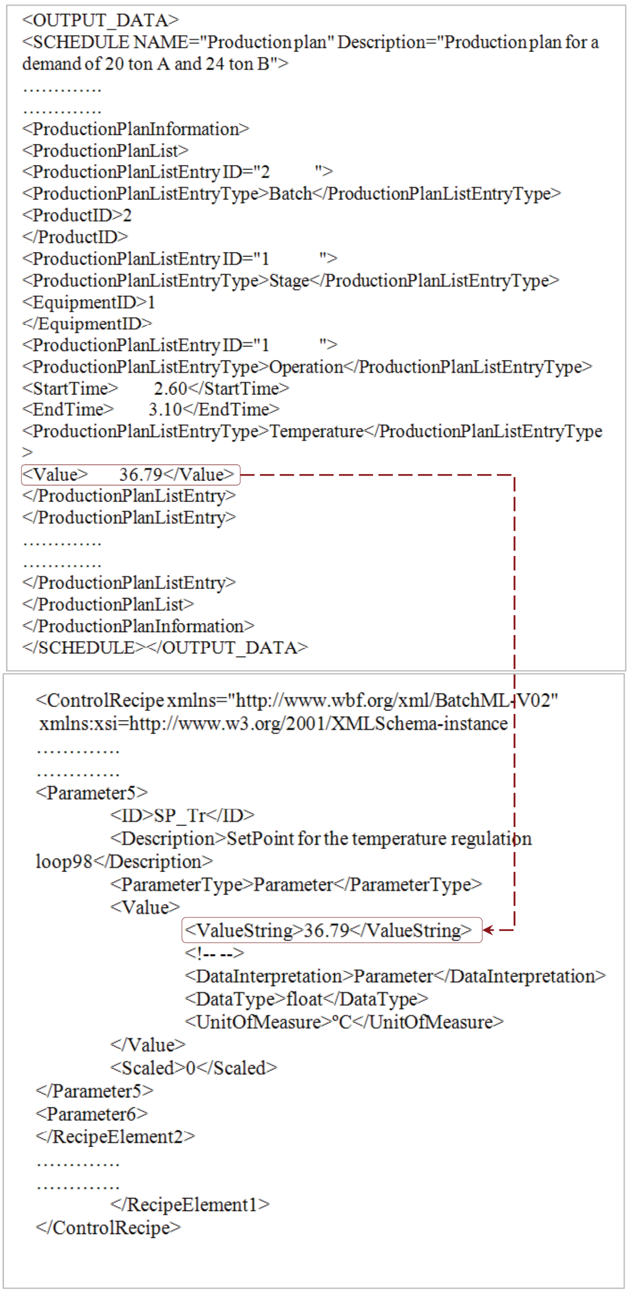
Results

In this case study no quantitative results are obtained but the operational procedure of the proposed framework is analyzed qualitatively.

The control task in the units modeled has been applied to bring the processing time data from the control level model through an special API of the ontology software, to the scheduling level model. In this way, variable values have been updated when the planing level asks for using them as input for scheduling. In order to succeed in this activity a link between data source from the external software variable and the real values captured by the control system in the process database has been created.

A first application of this ontology has been implemented to "close" the typical scheduling-fault analysis-rescheduling loop (control levels 0 to 3 of the Purdue reference model). The system is coordinated by an internal server acting as an information administrator that is consistent with the ontology structure. This can be achieved through the ontology web language-application program interface OWL-API, which is a Java interface and implementation for the W3C Web Ontology Language OWL.

The user interface interacts with a model of the ontology (batch processes domain) on the client side via a listener pattern. When the model needs to be filled with new



```
<OUTPUT_DATA>
<SCHEDULE NAME="Production plan" Description="Production plan for a
demand of 20 ton A and 24 ton B">
............
............
<ProductionPlanInformation>
<ProductionPlanList>
<ProductionPlanListEntry ID="2        ">
<ProductionPlanListEntryType>Batch</ProductionPlanListEntryType>
<ProductID>2
</ProductID>
<ProductionPlanListEntry ID="1        ">
<ProductionPlanListEntryType>Stage</ProductionPlanListEntryType>
<EquipmentID>1
</EquipmentID>
<ProductionPlanListEntry ID="1        ">
<ProductionPlanListEntryType>Operation</ProductionPlanListEntryType>
<StartTime>      2.60</StartTime>
<EndTime>      3.10</EndTime>
<ProductionPlanListEntryType>Temperature</ProductionPlanListEntryType
>
<Value>      36.79</Value>
</ProductionPlanListEntry>
</ProductionPlanListEntry>
............
............
</ProductionPlanListEntry>
</ProductionPlanList>
</ProductionPlanInformation>
</SCHEDULE></OUTPUT_DATA>
```

```
<ControlRecipe xmlns="http://www.wbf.org/xml/BatchML-V02"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
............
............
<Parameter5>
            <ID>SP_Tr</ID>
            <Description>SetPoint for the temperature regulation
loop98</Description>
            <ParameterType>Parameter</ParameterType>
            <Value>
                        <ValueString>36.79</ValueString>
                        <!-- -->
                        <DataInterpretation>Parameter</DataInterpretation>
                        <DataType>float</DataType>
                        <UnitOfMeasure>°C</UnitOfMeasure>
            </Value>
            <Scaled>0</Scaled>
</Parameter5>
<Parameter6>
</RecipeElement2>
............
............
            </RecipeElement1>
</ControlRecipe>
```

Figure 6.2: Temperature parameter following the ANSI/ISA-88 standard.

information, the Remote Procedure Call (RPC) module on the client side will invoke a request to the RPC module of the server, which interacts with the ontology and the Collaboration APIs to provide the requested data. The aim is to make it easier for knowledge engineers and experts to manage knowledge. The working process of the ontology modeling approach is shown in Figure 6.3, which illustrates the relation between different actors. In addition, it enables the user to reuse existing information from the modeling databases or to create and add new design information.

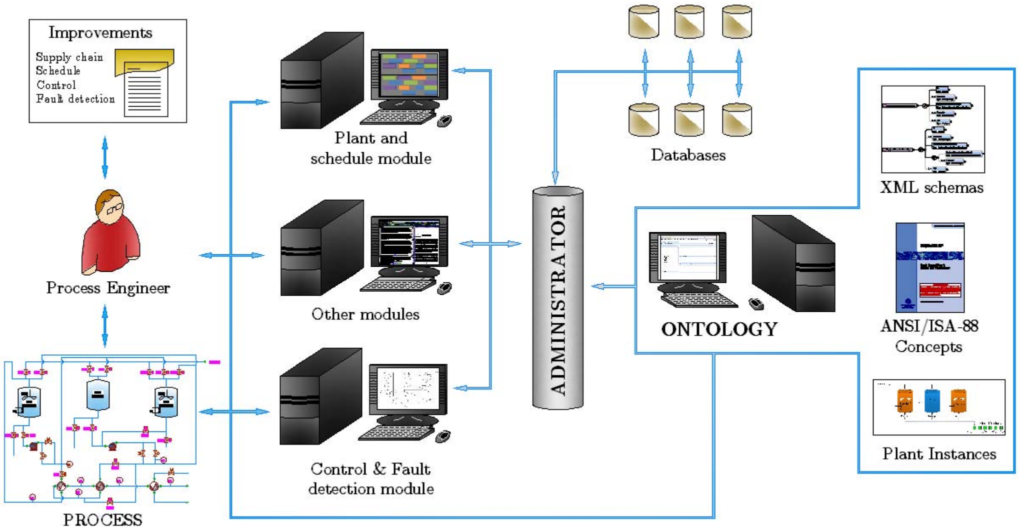


Figure 6.3: Ontology Model 1 approach for Case study 1.

6.2.2 Case study 2: Multi product batch plant - control and scheduling

Problem description

The case study consists of a multi product plant, manufacturing two products, i.e. A and B, through three process stages (Figure 6.4). Product batch sizes, optimal production times and product demands are shown in Tables 6.5 and 6.6. A single unit is available for each stage and unlimited intermediate storage policy is adopted. Product changeover time and cost are disregarded.

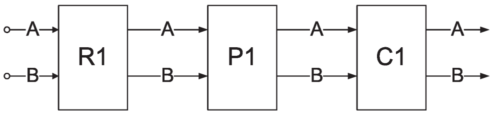


Figure 6.4: Plant flowsheet for the Case study 2.

The detailed piping and instrumentation diagram of the process plant is shown in Figure 6.5. Table 6.7 contains the identification of the different elements in the process recipe. These elements must be instantiated in the ontology as described in section 4.3.

The scheduling objective function, profit maximization (Eq. 6.1), includes the benefit of each batch and the operating cost, which is a function of the free process

Table 6.5: Product prices and lot sizes for Case study 2.

Product	Batch Benefit [m.u./batch]	Batch size [ton/batch]	Demand [ton]	Unitary energy cost [m.u./MWh]
A	30	5	20	90
B	40	6	24	90

Table 6.6: Recipe stage times [h].

Product	Stage 1		Stage 2		Stage 3	
	Unit	Time [h]	Unit	Time [h]	Unit	Time [h]
A	R1	0.5	P1	0.5	C1	0.5
B	R1	0.5	P1	0.8	C1	0.4

Table 6.7: Description of the process plant in Case study 2.

Concept	**Name**	**ID**	**State**	**Density**	**Capacity**
Product	End_Product A	LP00A	liquid		
	End_Product B	LP00B	liquid		
	Residue 1	R001	liquid		
	Residue 2	R002	liquid		
	By products	BP100	liquid		
Unit	Reactor	P-110			10 m^3
	Separator	D-120			10 m^3
	Separator	D-130			8 m^3
Stage	Reaction				
	Separation1				
	Separation2				
Resource	Raw Material A	RM001	liquid	1000kg/m^3	
	Raw Material B	RM002	liquid	1100kg/m^3	
	Raw Material C	RM003	liquid	1200kg/m^3	
Control Element	Electro Valve 1	J-112			
	Electro Valve 2	J-113			
	Electro Valve 3	J-122			
	Electro Valve 4	J-123			
	Electro Valve 5	J-124			
	Electro Valve 6	J-132			
	Electro Valve 7	J-133			
	Electro Valve 8	J-134			
Pumps	Pump 1	K-111			
	Pump 2	K-121			
Data Bases	Production Planning	PPDB		lot size	
	Production Plant Planning	PPPDB		Production costs	
				Demand	
				Price	
	Production	PDB		Operational times	
	control	CDB		temperature	
				time	
				reaction	

variables and is related to the processing time as a result. Specifically, linear functions (Eqs. 6.2 and 6.3) are used to calculate the operating costs depending on the required processing time; they are approximated from experimental data (i.e. simulation data from the rigorous process model) so that they can be easily included in the scheduling objective function.

$$z^{profit} = SellingPrice - OperatingCost - RawMaterialCost \tag{6.1}$$

$$OperatingCost_A = 32.6260 - 15.5879 \cdot ProcessingTime \tag{6.2}$$

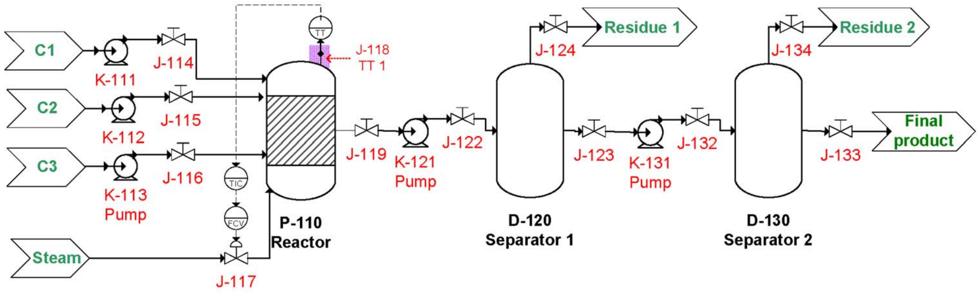


Figure 6.5: Piping and instrumentation diagram for Case study 2.

$$OperatingCost_B = 42.7954 - 27.6950 \cdot ProcessingTime \tag{6.3}$$

The first stage dynamics correspond to an isothermal reaction system with second order kinetics for both products A and B in a continuous stirred tank reactor. The higher temperature in this stage, the lower production time and higher energy cost. Basic control system in unit R1 is composed by a feedback temperature controller. The operation in this stage can be adapted to the scheduling level as mentioned, so that flexible processing times and operating costs can be obtained. This means that set point values are decided at the scheduling level instead of being decided at the process level, at inter-batch control level or by the supervision system.

Results

The re-usability of the ontology, that is its capacity to be applied to any process plant, is demonstrated by the successful instantiation of this case study. Specifically, the instantiation of the ontology contains 52 instances. Thus, the ontology provides the complete semantic model of the whole plant according to the batch process management model described in the ANSI/ISA-88. As a result, the same representation of the process is available to the various expert systems contained in the software, which hold every decision making function, namely programming, control and case-based reasoning, along with information needed to perform such tasks.

In turn, the informatics systems receive the results of the decisions taken out, and store such information in appropriate databases. In all cases, the informatics systems work supported by the ontological representation of the process. Hence, the ontology is regarded as usable, because the plant knowledge can be approached by several applications.

For example, when defining the plant equipment in the different functions, the scheduling level only requires a shallow level of detail (Figure 6.4), so the informatics system provides it with the "Unit" elements according to the ANSI/ISA-88 (P-110, D-120, P-130 from Figure 6.5); whereas the control level needs a deeper degree of description, and the informatics system provides it with all the elements up to "Equipment Modules" level (K-111, J-114, K-112 and so on, from Figure 6.5). On the whole, a single unified process representation of the plant can build different problem instances and be used to manage the information in spite of the specific problem that needs to be solved at each moment.

Next, two solution procedures for the scheduling problem are presented to illustrate the capabilities of the integrated information environment, depending on the available plant knowledge in the historical plant database. Both of them take advantage of the ontology usability and more efficient communication capabilities, allowing sharing of information between different decision levels with different degree of detail and time scales.

The first scheduling procedure (Figure 6.6), which includes rescheduling action when faults are detected at the process level, underlines improvements in reaction to incidences of both decision levels. On the other hand, the usability of the overall knowledge constructed through the information framework plant model is exploited in the second scheduling procedure (Figure 6.7), since previously implemented operating plans are used. As a result, the repetition of solving identical optimization problems is avoided and computing time and effort are saved.

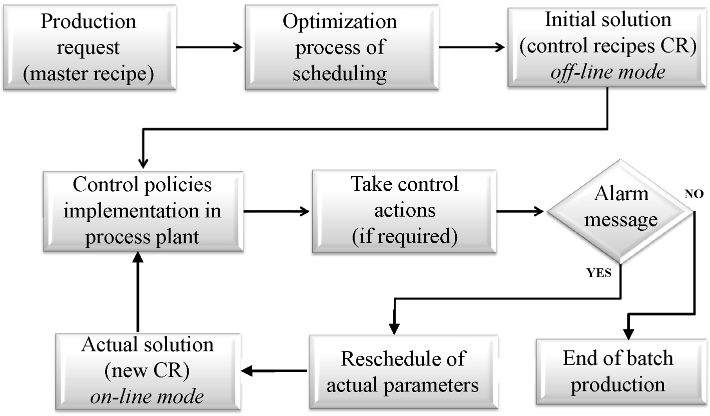

Production request (master recipe)
Optimization process of scheduling
Initial solution (control recipes CR) *off-line mode*
Control policies implementation in process plant
Take control actions (if required)
Alarm message
NO
YES
Actual solution (new CR) *on-line mode*
Reschedule of actual parameters
End of batch production

Figure 6.6: Diagram flow of the solution procedure exploiting the semantic representation to react against incidences.

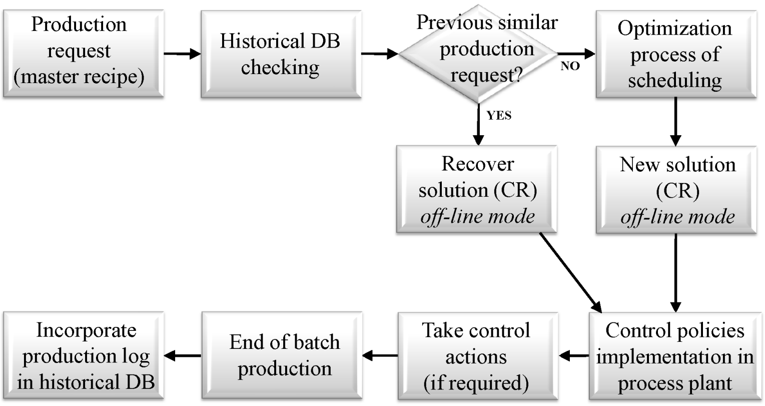

Production request (master recipe)
Historical DB checking
Previous similar production request?
NO
YES
Optimization process of scheduling
Recover solution (CR) *off-line mode*
New solution (CR) *off-line mode*
Incorporate production log in historical DB
End of batch production
Take control actions (if required)
Control policies implementation in process plant

Figure 6.7: Diagram flow of the solution procedure exploiting ontology usability.

Solution without previous plant knowledge

A solution procedure is proposed to solve the case study when no historical knowledge is considered (Figure 6.6). When a production order (Production request - master recipe, according to Figure 6.6) enters the production decision-level, the informatics application, described in Section 4.3.4, poses the problem for the scheduling function according to the incoming production orders and the plant representation contained in the ontological model. Therefore, the information necessary for building the scheduling model is automatically transferred to the optimizer.

Next, the scheduling function calculates the batch sequencing, allocation and timing to be executed by the process control level (Optimization process of scheduling, according to Figure 6.6). As a result, a production schedule is sent back to the informatics application and interpreted by the informatics application according to the ontological model. Therefore, the informatics system stores the data in the corresponding place of the data base. The Gantt chart of the solution is shown in Figure 6.8a.

Next, the informatics system builds the control recipe according to the production schedule and based on the whole process plant description of the ontological instantiation (Initial solution - control recipes off-line mode, according to Figure 6.6). Such control recipe is used by the control function to perform the optimized schedule for all the units at the control level (Control policies implementation in process plant, according to Figure 6.6).

Therefore, the scheduling solution, which is given through a set of batch-oriented control recipes (Figure 6.9a), is correlated to a set of equipment-oriented control recipes (Figure 6.9b) to be implemented (Take control actions if required, , according to Figure 6.6). Their correspondence is defined by the coordination control.

As a result, the simulation of the procedural control, basic control and process is carried out in Matlab as explained in Section 5.1. Thus, an exception alarm system is implemented in order to warn the scheduling level about production delays in real-time (Alarm message, according to Figure 6.6). In such cases, the real plant status is sent from the control function to the informatics application, which accordingly updates the database based on the plant description of the ontological model and the actual plant status.

Next, the informatics system poses the new problem with the updated information so that the scheduling function can build its model and optimize the new scheduling solution. Therefore, the solution of the processing requests is recalculated at the scheduling level having into account the current state of the process (Reschedule with actual parameters and Actual solution, according to Figure 6.6).

In this particular case study, an unexpected event has been introduced at time 1:30h: unit R1 (P-110) suffers a breakdown during the manufacturing of the fourth batch (second batch of product B), which has to be dismissed. For this reason, the control system sends an alarm signal and rescheduling is started. The first action can only be taken from the fifth batch on. The resulting scheduling is shown in Figure 6.8b. New actions are taken in the reactor control set points, and at the end, all required batches are served except for one batch of product A.

On the whole, the integrated framework is responsible for organizing and transferring the information and process constraints between the control level (for example, foreseen time of unit availability and processed orders) and the scheduling level (which defines not only the batch operation model but also temperature set-points). The batch process model behind the proposed information environment is based on ANSI/ISA-

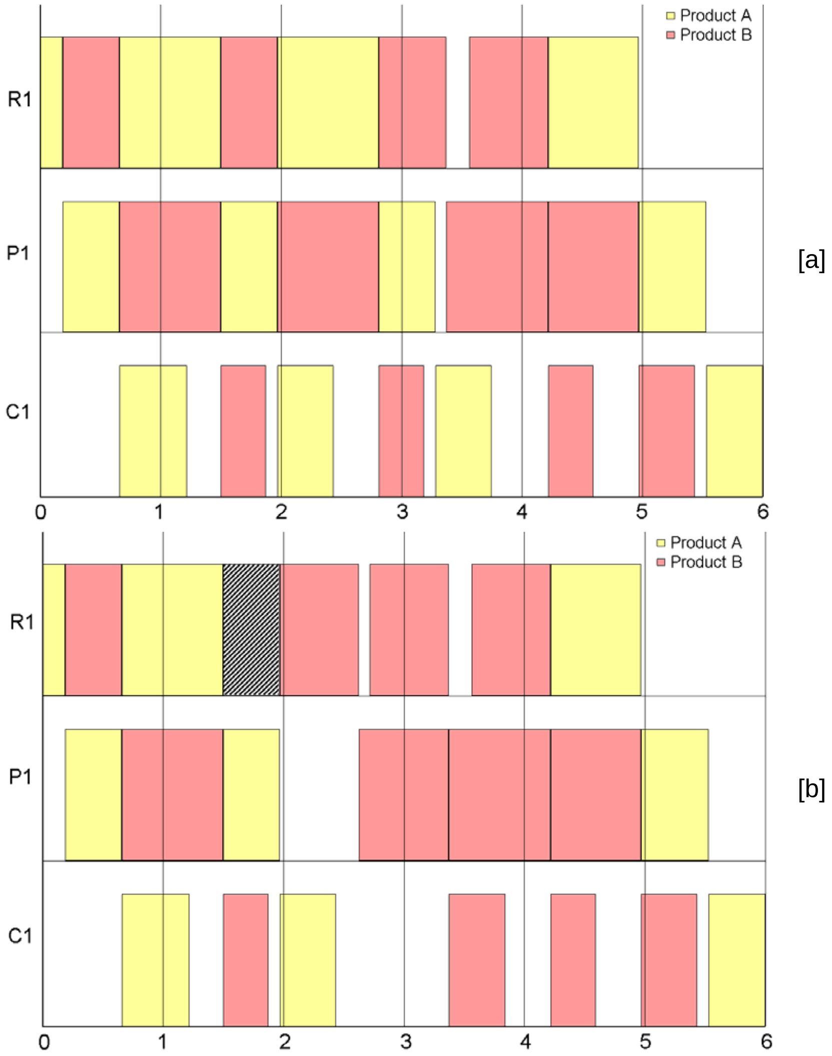


Figure 6.8: Gantt charts for Case study 2: a) initial solution and b) solution after the rescheduling action (the breakdown in unit R1 is shown shaded).

88 standard, which guarantees correspondence between the information of the several levels. Moreover, re-usability and communication efficiency in the provided integration framework can be exploited. In addition, the scheduling level eventually reschedules the orders; which demonstrates that high and effective reactivity to incidences can be achieved by the global system.

Solution with previous plant knowledge

The second solution approach (Figure 6.7) takes advantage of the ontology usability with the application of a Case-Based Reasoning (CBR). In this case, when a production order enters informatics application (Production request, according to Figure 6.7),

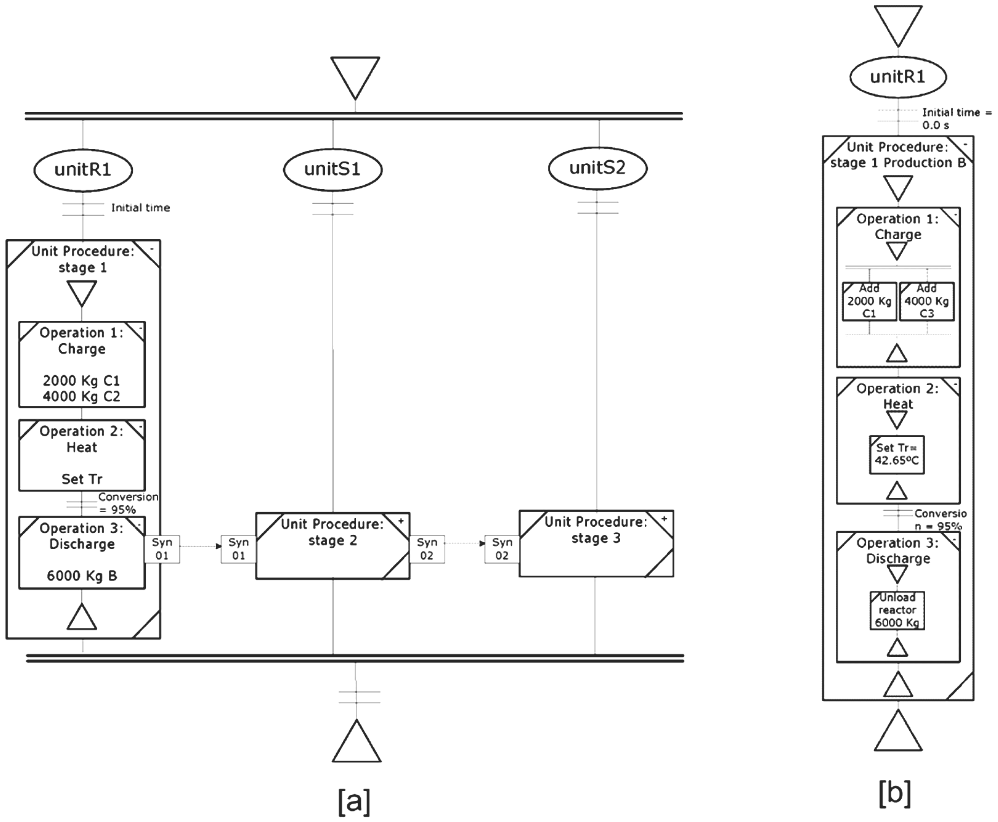


Figure 6.9: Batch operation models: a) batch-oriented recipe and b) equipment-oriented recipe.

an informatics system based on the ontological representation of the plant adapts the request for the interpretation of the CBR, which checks if any previous order fully or partly coincides with the current one (Historical DB checking/Previous similar production request, according to Figure 6.7). In such case, the scheduling optimizer is not required for the calculation of the whole batch sequencing, allocation and timing; instead, the scheduling solutions stored in the historical database are used (Recover solution off-line mode, according to Figure 6.7). Therefore, the stored control recipes are sent to the control function, which implements the production schedule (Control policies implementation in process plant and Take control actions, according to Figure 6.7). As a result, information can be directly reused, and the usability of the proposed ontological framework is successfully exploited. The final production schedule implementation must be incorporated in the historical database (Incorporate production log in historical DB, according to Figure 6.7).

6.2.3 Model remarks

This model represents the first approach toward the support for the integration, not just communication, of different software tools applicable to the management and exploitation of plant database information, resulting into an enhancement of the entire process management structure.

However the application of the first model to the previous case studies has opened new challenges which are to be included in the re-planing phase for a second model. Specifically, the master recipe definition in Model 1 is a black box which does not contain enough level of detail regarding the representation that can be achieved with an ontological model, for the complete integration of the decision levels and its future exploitation. For this reason, Model 2 has been developed in a higher detail in order to incorporate other functionalities of the scheduling level. Furthermore it has been proved that the re-planning phase in the methodology proposed improves the current ontologies development methodologies by including a continuous quality tool, the PDCA cycle. From this point of view, this model represents an ongoing successful effort to improve ontology process development in a friendly way.

In addition, this ontology opens the way for achieving successful flexible control in adapting and recognizing different elements found through the hierarchy models that are associated with manufacturing multilevel control systems.

Finally, it has been proved the adequacy of an ontology as a means for sharing information about a general model for different problem representations. As a result, it deals with the problem of integration, standardization and compatibility of heterogeneous modeling systems.

6.3 Model 2: Complete master recipe definition

This model is the result of the re-planning phase described in chapter 4. It improves the previous one by the addition of the elements contained in the master recipe detailed description, according to the model proposed by the world batch forum (WBF) (Brandl and Emerson, 2003). In this way new classes, properties and axioms have been defined in order to extract the planning and scheduling knowledge involved in the master recipe creation. For example, the specification of recipe elements, procedure logics, equipment requirements, header, formula, batch sizes, links, etc leads to the direct exploitation by the users decision tools.

This model comprises 129 classes, 73 axioms, and 130 properties. It is noteworthy to mention that the number of object properties has decreased compared to the previous model because although new properties have been added, some inverse properties relationships have been debugged. As a main feature of this new ontology, the connection of static data (structural related classes as equipment capacities) and dynamic data (operational related as demand in a time period) has been improved by the fact that the classes related to these elements have been included in the new model and the xml code is avoided.

This model has been applied to two different process plants for checking capabilities of the ontology to deal with both multi product and multi purpose structured scheduling problems.

6.3.1 Case Study 3: Multi product batch plant - scheduling

Problem description

The plant described in Case study 2 is revisited in this case study, giving emphasis to the master recipe definition. It is important to note that a production time horizon of 6.5 hours is considered, and fixed processing times are considered for stage 1.

Results

The re-usability of the proposed ontological model has been demonstrated through the instantiation of this case study. Specifically a total of 151 instances represent all the necessary problem features. The number of instances have been increased three times compared to Model 1 because of the detail of the master recipe which makes the model more complete (more classes). The total time for instantiating this case study consists of 5400s. The reasoner needs 0.49 sCPU to check the consistency of the inferred instances. In this instantiation of physical and procedural model, equipment requirement, processing times, costs, mass balances, raw material, and other elements have been referenced from the master recipe. For illustrating purposes, Figure 6.10 contains the instances derived from the master recipe of product A.

Table 6.8 contains all the instances of this case study for each class. As defined in Table 5.2, the instances of specific classes are required by the scheduling function in order to implement the scheduling optimization. Therefore, with the instances defined in Table 6.8 and the properties within the model, the scheduling problem can be fulfilled and optimized.

Table 6.8: Instances of each class for Case study 3.

Class	Instances
Administrator	-
Allocation	-
Arbitration	-
Area	-
BasicControl	-
Batch	-
BatchControl	-
BatchProcess	Separation_liquid;
BatchProduction	-
BatchSize	BatchSize_MasterRecipe_A; BatchSize_MasterRecipe_B;
ByProduct	-
Capacity	-
CleaningTime	-
ContinuosProcess	-
ControlActivities	-
ControlFunctions	-
ControlModule	ControlModule_ElectroValves; ControlModule_Pumping; ControlModule_Temperatur; ControlModule_none;
ControlRecipe	-
CoordinationControl	-
Demand	Demand_A_May; Demand_B_May;
DiscretPartProcess	-
DominianConcepWCW	-
EconomicResourses	Pesos;
EndProduct	EndProduct_A; EndProduct_B;
EnergeticResourses	electricity;
Enterprise	-
EquipmentControl	-
EquipmentEntity	-

EquipmentModule	ElectroValve1; ElectroValve2; ElectroValve3; ElectroValve4; ElectroValve5; ElectroValve6; ElectroValve7; ElectroValve8; Pump1; Pump2; Pump3; Resistence;
EquipmentOperation	-
EquipmentPhase	-
EquipmentProcedure	-
EquipmentRequirement	Equipment_1; Equipment_2; Equipment_3;
EquipmentUnitProcedure	-
Formula	Formula_Product_B; Formula_product_A;
GaneralRecipe	-
GeneralInformation	-
Header	product_Recipe_A; product_Recipe_B;
HumanResourses	-
ID	-
IDByProduct	-
IDEndProduct	ID_FP00A; ID_FP00B;
IDEquipment	ID_D-120; ID_D-130; ID_P-110;
IDEquipmentModule	ID_J-112; ID_J-113; ID_J-122; ID_J-123; ID_J-124; ID_J-132; ID_J-133; ID_J-134; ID_K-111; ID_K-112; ID_K-113; ID_R-110;
IDIntermediate	IN001A; IN001B; IN002A; IN002B;
IDLink	ID_LA1; ID_LA2; ID_LB1; ID_LB2;
IDLogic	LogicID_A; LogicID_B;
IDMaterial	-
IDProcessStage	PS00A1; PS00A2; PS00A3; PS00B1; PS00B2; PS00B3;
IDRawMaterial	ID_RM001; ID_RM002; ID_RM003;
IDRecipe	MR00A; MR00B;
IDRecipeElement	REA1S1; REA1S2; REA1S3; REB1S1; REB1S2; REB1S3;
IDRecipeElementType	-
IDRecipeOperation	-
IDRecipePhase	-
IDResidue	ID_R001; ID_R002;
IndustrialManufacturingProcess	-
Information	-
Intermediate	Intermediate_1A; Intermediate_1B; Intermediate_2A; Intermediate_2B;
InventoryEconomicResources	-
InventoryEnergeticResources	-
InventoryHumanResources	-
InventoryManagement	-
InventoryMaterialResources	InventoryMaterialResources_C1; InventoryMaterialResources_C2; InventoryMaterialResources_C3; InventoryMaterialResources_FPA; InventoryMaterialResources_FPB;
Lot	-
MasterRecipe	MasterRecipe_A; MasterRecipe_B;
MaterialResourses	-
MixingSpeed	-
Mode	-
Operation	-
P_Capacity	-
P_Materials	C1_A; C1_B; C2_A; C2_B; C3_A; C3_B; I1_A; I1_B; I2_A; I2_B; fpA_A; fpB_B; residue_1_A; residue_1_B; residue_2_A; residue_2_B;
P_MixingSpeed	-
P_Pressure	-
P_Temperature	reactor_temp_A; reactor_temp_B;
P_Time	PT_A_S1; PT_A_S2; PT_A_S3; PT_B_S1; PT_B_S2; PT_B_S3;
P_TransferRate	-
P_ValvePosition	-
Parameter	-
Phase	-
PhysicalModel	-
Pressure	-
ProceduralControl	-
ProceduralElement	-
ProceduralLink	link_1; link_2; link_3; link_4;
ProceduralLogic	ProceduralLogic_A; ProceduralLogic_B;
Procedure	-
Process	-
ProcessAction	-

ProcessCell	-
ProcessControl	-
ProcessInformation	-
ProcessingActivities	-
ProcessingTime	ProcessingTime_S1; ProcessingTime_S2; ProcessingTime_S3;
ProcessManagement	-
ProcessOperation	Reaction; Separation;
ProcessOutput	-
ProcessParameter	-
ProcessStage	ProcessStage_A1; ProcessStage_A2; ProcessStage_A3; ProcessStage_B1; ProcessStage_B2; ProcessStage_B3;
ProductionPolicy	Earliness; EnvironmentalImpact; Makespan; Profit; Tardiness;
ProductionProcess	-
ProductionRequirement	-
RawMaterial	RawMaterial_A; RawMaterial_B; RawMaterial_C;
RecipeElement	RecipeElement_A1; RecipeElement_A2; RecipeElement_A3; RecipeElement_B1; RecipeElement_B2; RecipeElement_B3;
RecipeManagement	-
RecipeOperation	-
RecipePhase	-
RecipeProcedure	-
RecipeProcess	-
RecipeType	-
RecipeUnitProcedure	-
Requeriments	-
Residue	Residue_1; Residue_2;
Resources	-
Restriction	-
Site	-
SiteRecipe	-
State	-
Temperature	reactor_temp;
Thing	-
Time	-
TranferRate	-
Unit	Reactor; Separator_1; Separator_2;
UnitProcedure	-
UnitResources	-
UnitSupervision	-
ValvePosition	-
WatingTime	-

Regarding the re-usability, results shown in Gantt chart in Figure 6.11 are obtained after the problem instantiation. Thus, the use of the ontology provides a smarter way to manage data, in contrast to traditional scheduling problem instantiations in which the problem is described for a particular case and cannot be shared with other applications or other optimization models.

6.3.2 Case study 4: Benchmark problem - Kondili scheduling

Problem description

The case study consists of a multi-purpose batch plant, which is a benchmark case study of the scheduling area (Kondili et al., 1993).The production process consist of five production tasks and nine states, namely three raw materials, two final products and four intermediates. Figure 6.12 shows the process flow sheet as well as the processing times and mass balances.

A single unit is available for each stage and unlimited intermediate storage policy is adopted. Product changeover time and cost are disregarded.

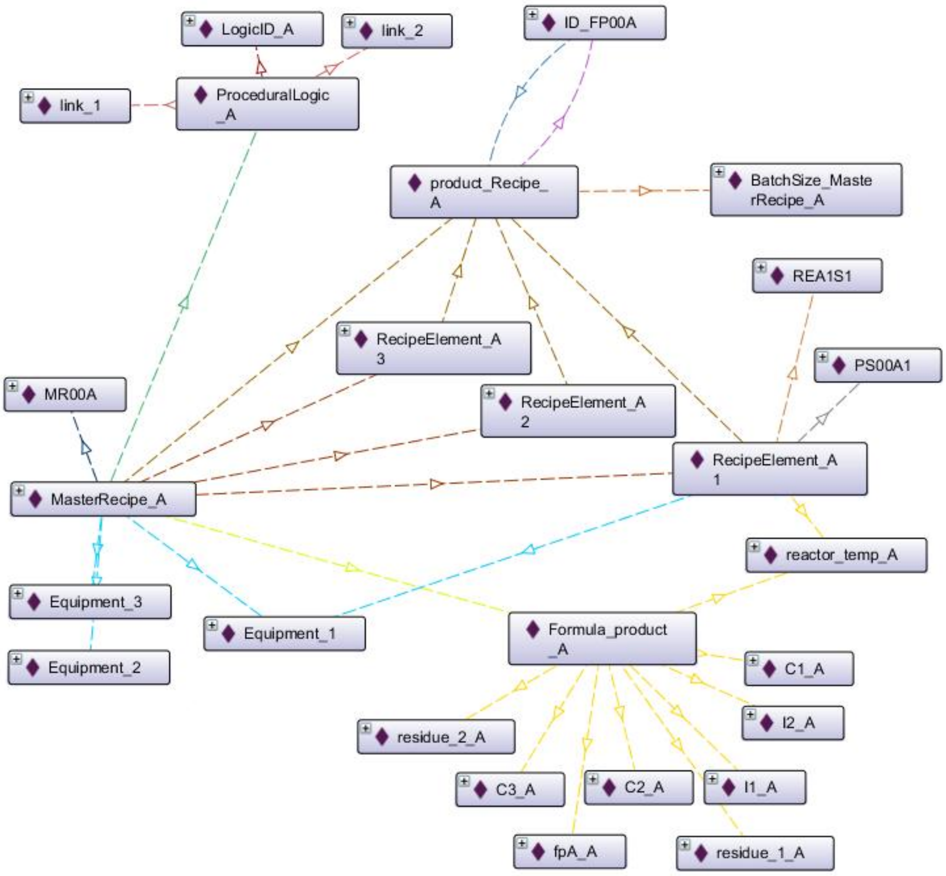


Figure 6.10: Instances derived from master recipe of product A in Case study 3.

The scheduling objective function consists of production maximization within 18 h time horizon.

Results

The re-usability of the proposed ontological model has been proved through the instantiation of the multi-purpose batch plant. Specifically a total of 111 instances represent all the necessary problem features. The reasoner needs 0.172 sCPU to check the consistency of the inferred instances. In this case, the production process has been divided in two main recipes. For illustrating purposes, Figure 6.13 contains the instances derived from the master recipes.

Table 6.10 contains all the instances of this case study for each class. As defined in Table 5.2, the instances of specific classes are required by the scheduling function in order to implement the scheduling optimization. Therefore, with the instances defined in Table 6.10 and the properties within the model, the scheduling problem can be fulfilled and optimized.

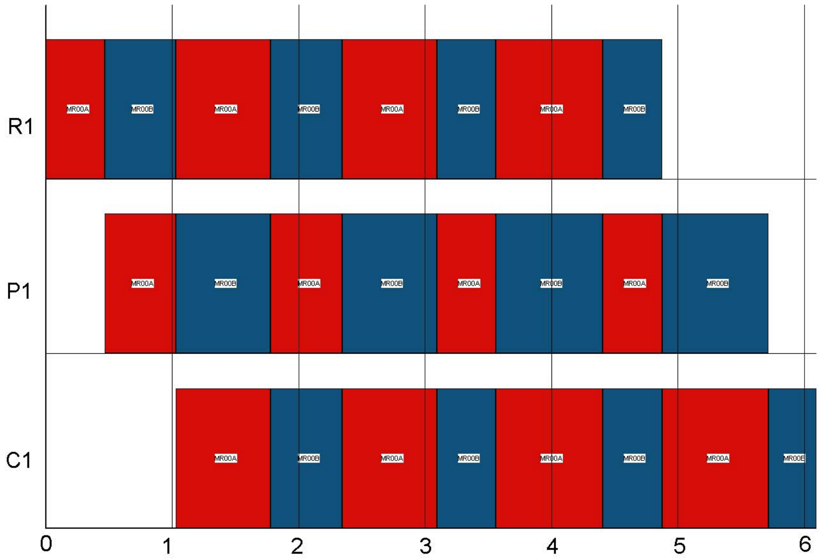


Figure 6.11: Gantt chart resulting from the optimization of Case study 3.

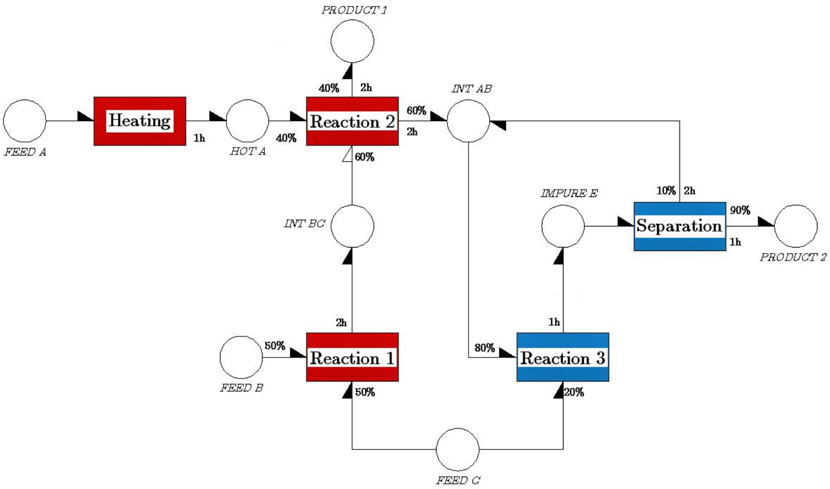


Figure 6.12: STN representation for the Case study 4.

Table 6.9: Instances of each class for Case study 4.

Class	Instances
Administrator	-
Allocation	-
Arbitration	-
Area	-
BasicControl	-
Batch	-
BatchControl	-
BatchProcess	-
BatchProduction	-
BatchSize	BatchSize_P1; BatchSize_P2;
ByProduct	-

Capacity	-
CleaningTime	-
ContinuosProcess	-
ControlActivities	-
ControlFunctions	-
ControlModule	HeatingReactorControlModule; ReactorControlModule; SeparationControlModule;
ControlRecipe	-
CoordinationControl	-
Demand	DemandProduct1; DemandProduct2;
DiscretPartProcess	-
DominianConcepWCW	-
EconomicResourses	-
EndProduct	Product1; Product2;
EnergeticResourses	-
Enterprise	-
EquipmentControl	
EquipmentEntity	-
EquipmentModule	HeatingReactorModule; ReactorModule; SeparationModule;
EquipmentOperation	-
EquipmentPhase	-
EquipmentProcedure	-
EquipmentRequirement	EquipmentRequirement_1; EquipmentRequirement_2; EquipmentRequirement_3; EquipmentRequirement_4;
EquipmentUnitProcedure	-
Formula	Formula_P1; Formula_P2;
GaneralRecipe	-
GeneralInformation	-
Header	Header_P1; Header_P2;
HumanResourses	-
ID	-
IDByProduct	-
IDEndProduct	FP001; FP002;
IDEquipment	HR; RR1; RR2; SR;
IDEquipmentModule	-
IDIntermediate	IN001; IN002; IN003; IN004;
IDLink	IDLink_1; IDLink_2; IDLink_3;
IDLogic	Log_P1; Log_P2;
IDMaterial	-
IDProcessStage	PS_P1_H; PS_P1_R1; PS_P1_R2; PS_P2_R1; PS_P2_S;
IDRawMaterial	RM001; RM002; RM003;
IDRecipe	R_P1; R_P2;
IDRecipeElement	RE_P1_S1; RE_P1_S2; RE_P1_S3; RE_P2_S1; RE_P2_S2;
IDRecipeElementType	-
IDRecipeOperation	-
IDRecipePhase	-
IDResidue	-
IndustrialManufacturingProcess	-
Information	-
Intermediate	HotA; ImpureE; IntAB; IntBC;
InventoryEconomicResources	-
InventoryEnergeticResources	-
InventoryHumanResources	-
InventoryManagement	-
InventoryMaterialResources	Inv_A; Inv_B; Inv_C;
Lot	-
MasterRecipe	MasterRecipe_P1; MasterRecipe_P2;
MaterialResourses	-
MixingSpeed	-
Mode	-
Operation	-
P_Capacity	-
P_Materials	AB_P1; AB_P2_In; AB_P2_Out; A_P1; BC_P1; B_P1; C_P1; C_P2; E_P2; HA_P1; P1; P2;
P_MixingSpeed	-
P_Pressure	-
P_Temperature	-

P_Time	P_Time_Heating_P1; P_Time_Reaction1_P1; P_Time_Reaction1_P2; P_Time_Reaction2_p1; P_Time_Separation_P2;
P_TransferRate	-
P_ValvePosition	-
Parameter	-
Phase	-
PhysicalModel	-
Pressure	-
ProceduralControl	-
ProceduralElement	-
ProceduralLink	Link_1; Link_2; Link_3;
ProceduralLogic	ProceduralLogic_P1; ProceduralLogic_P2;
Procedure	-
Process	-
ProcessAction	-
ProcessCell	-
ProcessControl	-
ProcessInformation	-
ProcessingActivities	-
ProcessingTime	ProcessingTime_Heating; ProcessingTime_Reaction1; ProcessingTime_Reaction2; ProcessingTime_Reaction3; ProcessingTime_Separation;
ProcessManagement	-
ProcessOperation	Heating; Reaction; Separation;
ProcessOutput	-
ProcessParameter	-
ProcessStage	ProcessStage_P1_Reaction1; ProcessStage_P1_Reaction2; ProcessStage_P1_heating; ProcessStage_P2_Reaction; ProcessStage_P2_Separation;
ProductionPolicy	Earliness; EnvironmentalImpact; Makespan; Profit; Tardiness;
ProductionProcess	-
ProductionRequirement	-
RawMaterial	FeedA; FeedB; FeedC;
RecipeElement	RecipeElement_P11; RecipeElement_P12; RecipeElement_P13; RecipeElement_P21; RecipeElement_P22;
RecipeManagement	-
RecipeOperation	-
RecipePhase	-
RecipeProcedure	-
RecipeProcess	-
RecipeType	-
RecipeUnitProcedure	-
Requeriments	-
Residue	-
Resources	-
Restriction	-
Site	-
SiteRecipe	-
State	-
Temperature	-
Thing	-
Time	-
TranferRate	-
Unit	HeatingReactor; Reactor1; Reactor2; Separation_Unit;
UnitProcedure	-
UnitResources	-
UnitSupervision	-
ValvePosition	-
WatingTime	-

Regarding the re-usability, results shown in the Gantt chart in Figure 6.14 are obtained after the problem instantiation. As in the previous case study, the use of the ontology provides a smart way to manage data. Additionally, this case demonstrates that different plant structures can be easily instantiated in the ontological model and

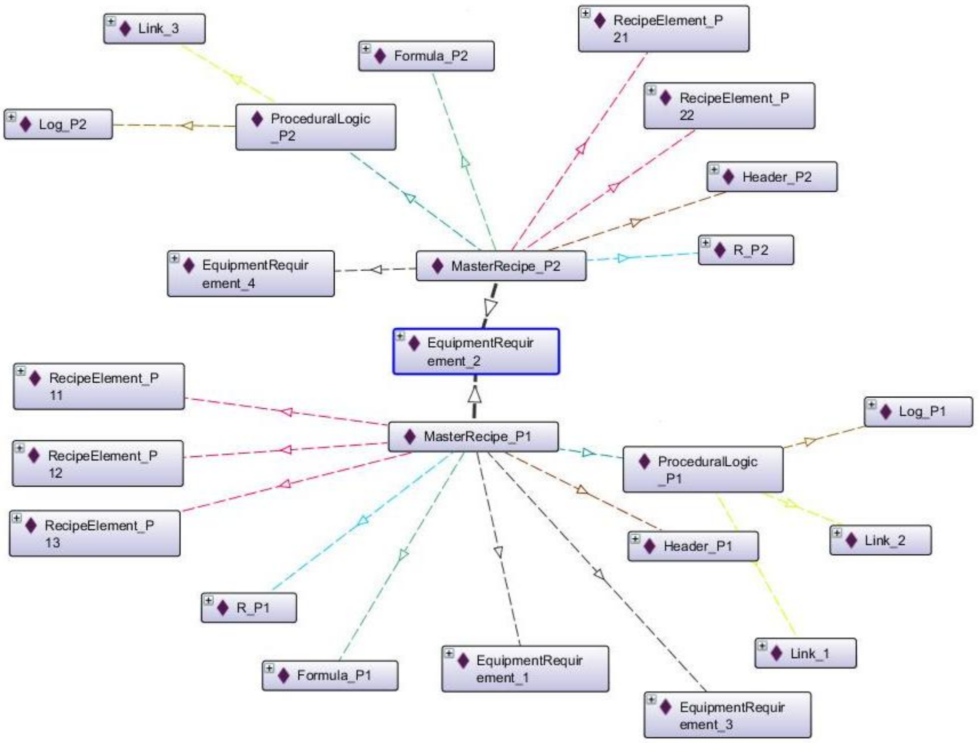


Figure 6.13: Instances derived from the master recipes in Case study 4.

automatically used by the informatics application to generate the necessary files to be used by the optimizer.

6.3.3 Case study 5: Integration - scheduling

Problem description

This case study comprises the model integration of the two previous production plants. Specifically, the re-usability of the ontology has allowed to directly instantiate this case study based on the previous ones. Even more, in this case study different features regarding plant configuration (equipment, storage capacity, raw materials) can be turned off or on as convenient. Therefore, Figure 6.15 presents the two STN representations corresponding to the afore mention production lines. In this case, equipment resources are not shared. However, equipments could be used by any new recipe if equipment integration was pursued.

Results

The re-usability of the proposed ontological model has been proved by the instantiation of the two production lines. Specifically a total of 244 instances represent all the necessary problem features described previously in Section 5.2. The reasoner needs 1.42 sCPU to check the consistency of the inferred instances. For illustrating purposes, Figure 6.16 contains the instances derived from the master recipes.

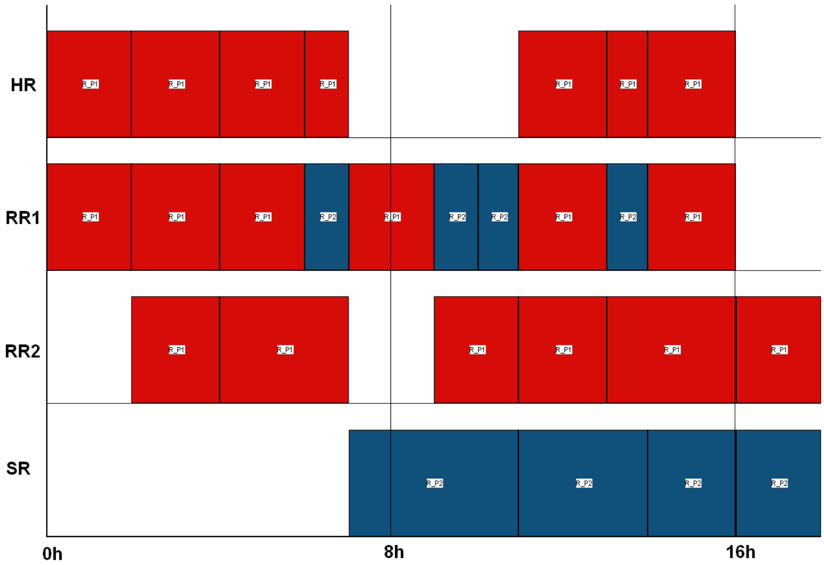


Figure 6.14: Gantt chart resulting from the optimization of Case study 4.

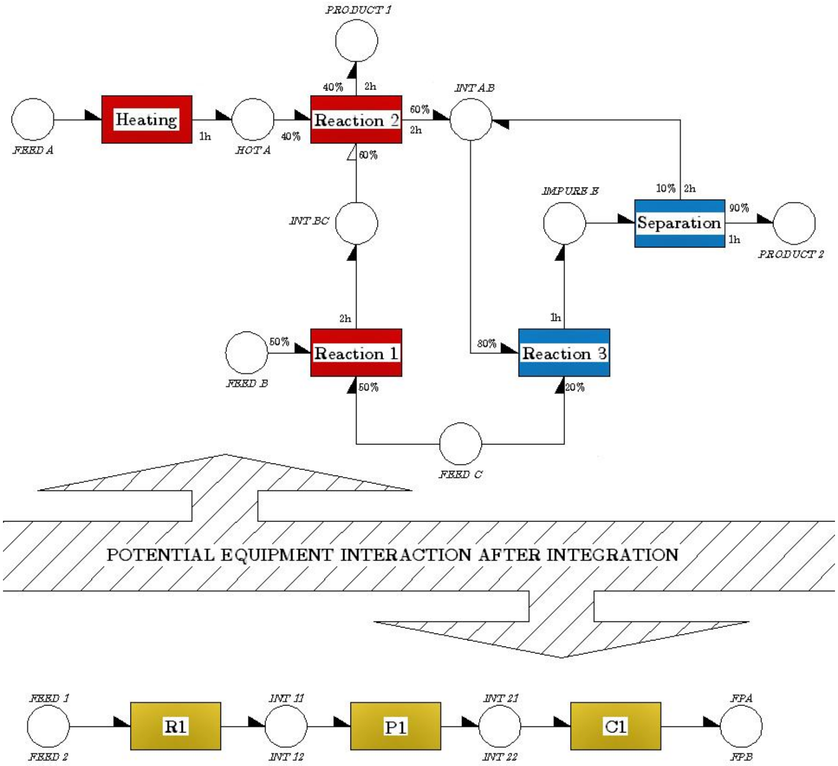


Figure 6.15: STN representation showing the potential integration of the two production lines for Case study 5.

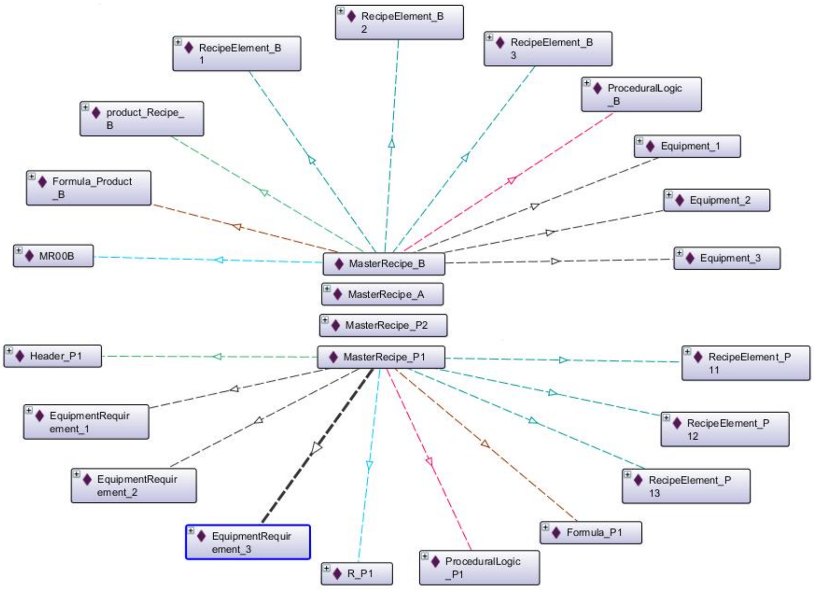


Figure 6.16: Instances derived from the master recipes in Case study 5.

Regarding the re-usability, results shown in Gantt chart in Figure 6.17 are obtained after the problem instantiation.

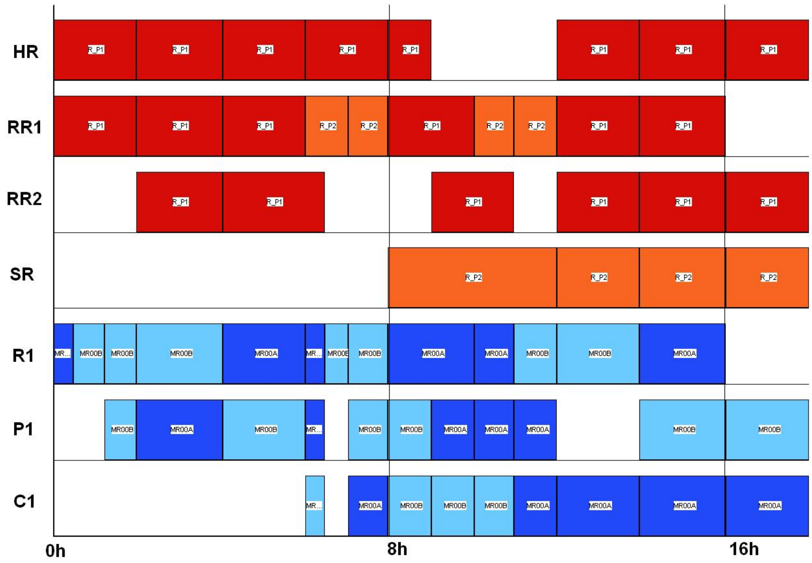


Figure 6.17: Gantt chart resulting from the optimization of Case study 5.

6.3.4 Model remarks

As demonstrated in previous examples, this model contributes to improve communication within the plant process environment, and represents a step forward to support the integration of different software tools applicable to the management and exploitation of plant database information, resulting into an enhancement of the entire scheduling function. Specifically, once the problem is instantiated, the modification of processing times, product demand or due date, resulting from plant redesign or new orders, is straight-forward and can be traced back to adequate databases.

Thus, the model enhances the way for achieving a successful scheduling decision-making supporting tool which adapts and recognizes the different elements found in the master recipe. Moreover, a general semantic framework is proposed, which is able to model any scheduling plant layout, proving its re-usability. Furthermore, it has been proved the ontology usability by its application to an optimization framework. As a whole, the main contributions of this environment and the model behind are re-usability, usability, higher efficiency in communication and coordination procedures.

In addition, it has been proved the adequacy of an ontology as a means for sharing information about a general model for different problem representations. As a result, it solves the problem of integration, standardization and compatibility of heterogeneous modeling systems. Even more, the response time for decision-making task could be reduced and better decisions adopted owing to faster availability of higher quality data and the improved visibility of the existing relationships between the scheduling function and other hierarchical levels functions.

Finally, given the advantages presented in this model the fact of including an additional decision level seems appropriate. For this reason the re-planning phase after checking Model 2 leads to the modeling of the SC level in the enterprise ontology model.

6.4 Model 3: Complete supply chain management

This model is the result of the second re-planning phase described in chapter 4. In this phase, the domain of the supply chain was taken in to account for its implementation in the ontological model as an expanded area for the complete enterprise integration. In fact this model corresponds to the one described in Chapter 4. The most important features defined are the information pieces found in the classes related to inventory, location, production & distribution and transportation management. In this way the supply chain management functions have been successfully modeled leading to the integration of the strategic level with the other decision levels already modeled.

This model comprises 182 classes, 64 axioms, and 155 properties. The number of the axioms has decreased as a result of the model debugging, which eliminated unnecessary or unused restrictions. The remarkable features of this model are described in chapter 4: the supply chain management, the supply chain design and retrofit, integration of all decision levels within a single model and the consequent improvement of the decision support task.

This model has been applied to a supply chain network problem presented by Lainez et al. (2009) based on a benchmark of the scheduling problem (Kondili et al., 1993). This case study demonstrates the re-usability of previous ontologies and the capability of integrating different levels of detail within a single model. Even more, the capability of delivering information quality at the right place in the right moment enhances the decision maker possibilities.

6.4.1 Case study 6: Benchmark problem - Lainez supply chain

Problem description

The case study is based on a supply chain network design-planning problem presented by Lainez et al. (2009). It consists of three suppliers, four potential locations for the processing sites and the distribution centers in a planning horizon of five annual periods (Figure 6.18). The production process fulfills the demand of six markets that entails two final products and one intermediate product.

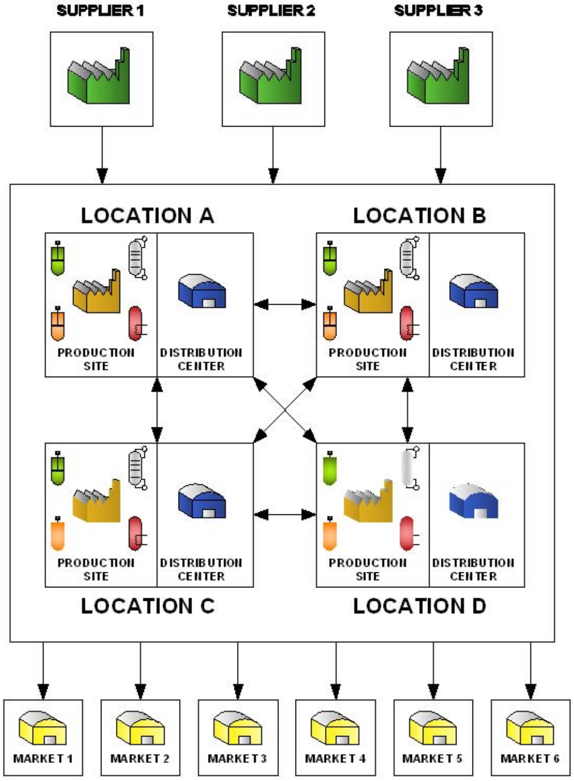


Figure 6.18: Supply chain structure of Case study 6.

Results

Quantitatively speaking, the problem representation in the proposed ontological framework results in 573 instances (Table 6.10). The reasoning time for the problem instances is 0.922 sCPU in a successful compilation.

It is important to mention that each possible site is fully represented in the ontology. Each production plant (site) may contain a set of four equipment technologies as presented by Kondili et al. (1993), a benchmark problem for the scheduling of batch process industries. The production process is described in Case study 4 (Figure 6.12). Specifically, each site is described by 111 instances, which may be adequately used to make operational decisions.

The results of the optimization model cited in Section 5.3 are identical to those reported in the original paper (Lainez et al., 2009). Furthermore, the previous results can be dated back to the ontological model for further exploitation by the other decision levels, such as the operational system of each site. This can be achieved by automatically updating the databases with the resulting optimization data.

Table 6.10: Instances of each class for Case study 6.

Class	Instances
Administrator	-
Allocation	-
Arbitration	-
Area	Area_127; Area_251; Area_3; Area_Kondili;
BasicControl	-
Batch	-
BatchControl	-
BatchProcess	-
BatchProduction	-
BatchSize	BatchSize_117; BatchSize_202; BatchSize_241; BatchSize_326; BatchSize_365; BatchSize_78; BatchSize_P1; BatchSize_P2;
ByProduct	-
Capacity	-
CleaningTime	-
ContinuosProcess	-
ControlActivities	-
ControlFunctions	-
ControlModule	ControlModule_13; ControlModule_137; ControlModule_143; ControlModule_19; ControlModule_214; ControlModule_261; ControlModule_267; ControlModule_338; ControlModule_90; HeatingReactorControlModule; ReactorControlModule; SeparationControlModule;
ControlRecipe	-
CoordinationControl	-
Demand	Demand_M1IntABT1; Demand_M1P1T1; Demand_M1P2T1; Demand_M2IntABT1; Demand_M2P1T1; Demand_M2P2T1; Demand_M3IntABT1; Demand_M3P1T1; Demand_M3P2T1; Demand_M4IntABT1; Demand_M4P1T1; Demand_M4P2T1; Demand_M5IntABT1; Demand_M5P1T1; Demand_M5P2T1; Demand_M6IntABT1; Demand_M6P1T1; Demand_M6P2T1;
DirectDistributionCenterCost	-
DirectManufacturingCost	-
DiscretPartProcess	-
Distance	-
DistributionCenter	DistributionCenter_1; DistributionCenter_2; DistributionCenter_3; DistributionCenter_4;
DistributionCenterPlacement	LA; LB; LC; LD;
DominianConcepWCW	-
EconomicResourses	-
EndProduct	Product1; Product2;
EnergeticResourses	-
Enterprise	-
EquipmentControl	-
EquipmentEntity	-
EquipmentMaintenance	-

EquipmentModule	EquipmentModule_133; EquipmentModule_142; EquipmentModule_18; EquipmentModule_213; EquipmentModule_257; EquipmentModule_266; EquipmentModule_337; EquipmentModule_89; EquipmentModule_9; HeatingReactorModule; ReactorModule; SeparationModule;
EquipmentOperation	-
EquipmentPhase	-
EquipmentProcedure	-
EquipmentRequirement	EquipmentRequirement_1; EquipmentRequirement_130; EquipmentRequirement_139; EquipmentRequirement_145; EquipmentRequirement_15; EquipmentRequirement_2; EquipmentRequirement_21; EquipmentRequirement_210; EquipmentRequirement_254; EquipmentRequirement_263; EquipmentRequirement_269; EquipmentRequirement_3; EquipmentRequirement_334; EquipmentRequirement_4; EquipmentRequirement_6; EquipmentRequirement_86;
EquipmentUnitProcedure	-
Facility	-
FacilityInvestmentCost	FacilityInvestmentCost_DC1; FacilityInvestmentCost_DC2; FacilityInvestmentCost_DC3; FacilityInvestmentCost_DC4; FacilityInvestmentCost_Site1; FacilityInvestmentCost_Site2; FacilityInvestmentCost_Site3; FacilityInvestmentCost_Site4;
Formula	Formula_147; Formula_217; Formula_23; Formula_271; Formula_341; Formula_93; Formula_P1; Formula_P2;
GaneralRecipe	-
GeneralInformation	-
GeographicPlacement	-
GoodsandSevices	-
Header	Header_116; Header_201; Header_240; Header_325; Header_364; Header_77; Header_P1; Header_P2;
HumanResourses	-
ID	-
IDArea	-
IDByProduct	-
IDEndProduct	FP001; FP002; IDEndProduct_157; IDEndProduct_221; IDEndProduct_281; IDEndProduct_33; IDEndProduct_345; IDEndProduct_97;
IDEquipment	HR; IDEquipment_11; IDEquipment_132; IDEquipment_135; IDEquipment_141; IDEquipment_17; IDEquipment_212; IDEquipment_256; IDEquipment_259; IDEquipment_265; IDEquipment_336; IDEquipment_8; IDEquipment_88; RR1; RR2; SR;
IDEquipmentModule	-
IDIntermediate	IDIntermediate_101; IDIntermediate_160; IDIntermediate_163; IDIntermediate_166; IDIntermediate_225; IDIntermediate_284; IDIntermediate_287; IDIntermediate_290; IDIntermediate_349; IDIntermediate_36; IDIntermediate_39; IDIntermediate_42; IN001; IN002; IN003; IN004;
IDLink	IDLink_1; IDLink_114; IDLink_192; IDLink_199; IDLink_2; IDLink_238; IDLink_3; IDLink_316; IDLink_323; IDLink_362; IDLink_68; IDLink_75;
IDLogic	IDLogic_115; IDLogic_200; IDLogic_239; IDLogic_324; IDLogic_363; IDLogic_76; Log_P1; Log_P2;
IDMaterial	-
IDProcessCell	IDProcessCell_122; IDProcessCell_246; IDProcessCell_370; KON001;
IDProcessStage	IDProcessStage_105; IDProcessStage_109; IDProcessStage_183; IDProcessStage_188; IDProcessStage_194; IDProcessStage_229; IDProcessStage_233; IDProcessStage_307; IDProcessStage_312; IDProcessStage_318; IDProcessStage_353; IDProcessStage_357; IDProcessStage_59; IDProcessStage_64; IDProcessStage_70; PS_P1_H; PS_P1_R1; PS_P1_R2; PS_P2_R1; PS_P2_S;

IDRawMaterial	IDRawMaterial_154; IDRawMaterial_173; IDRawMaterial_180; IDRawMaterial_278; IDRawMaterial_297; IDRawMaterial_30; IDRawMaterial_304; IDRawMaterial_49; IDRawMaterial_56; RM001; RM002; RM003;
IDRecipe	IDRecipe_146; IDRecipe_216; IDRecipe_22; IDRecipe_270; IDRecipe_340; IDRecipe_92; R_P1; R_P2;
IDRecipeElement	IDRecipeElement_119; IDRecipeElement_121; IDRecipeElement_204; IDRecipeElement_206; IDRecipeElement_208; IDRecipeElement_243; IDRecipeElement_245; IDRecipeElement_328; IDRecipeElement_330; IDRecipeElement_332; IDRecipeElement_367; IDRecipeElement_369; IDRecipeElement_80; IDRecipeElement_82; IDRecipeElement_84; RE_P1_S1; RE_P1_S2; RE_P1_S3; RE_P2_S1; RE_P2_S2;
IDRecipeElementType	-
IDRecipeOperation	-
IDRecipePhase	-
IDResidue	-
IDSite	-
InboundTransportationCost	-
IndirectDistributionCenterCost	-
IndirectManufacturingCost	-
IndustrialManufacturingProcess	-
Information	-
InstalationCost	InstalationCost_12; InstalationCost_136; InstalationCost_138; InstalationCost_14; InstalationCost_144; InstalationCost_20; InstalationCost_215; InstalationCost_260; InstalationCost_262; InstalationCost_268; InstalationCost_339; InstalationCost_91; InstalationCost_HeaterReactor; InstalationCost_Reactor1; InstalationCost_Reactor2; InstalationCost_Separator;
InterfacilityTransportationCost	InterfacilityTransportationCost_LA_LB; InterfacilityTransportationCost_LA_LC; InterfacilityTransportationCost_LA_LD; InterfacilityTransportationCost_LA_M1; InterfacilityTransportationCost_LA_M2; InterfacilityTransportationCost_LA_M3; InterfacilityTransportationCost_LA_M4; InterfacilityTransportationCost_LA_M5; InterfacilityTransportationCost_LA_M6; InterfacilityTransportationCost_LB_LC; InterfacilityTransportationCost_LB_LD; InterfacilityTransportationCost_LB_M1; InterfacilityTransportationCost_LB_M2; InterfacilityTransportationCost_LB_M3; InterfacilityTransportationCost_LB_M4; InterfacilityTransportationCost_LB_M5; InterfacilityTransportationCost_LB_M6; InterfacilityTransportationCost_LC_LD; InterfacilityTransportationCost_LC_M1; InterfacilityTransportationCost_LC_M2; InterfacilityTransportationCost_LC_M3; InterfacilityTransportationCost_LC_M4; InterfacilityTransportationCost_LC_M5; InterfacilityTransportationCost_LC_M6; InterfacilityTransportationCost_LD_M1; InterfacilityTransportationCost_LD_M2; InterfacilityTransportationCost_LD_M3; InterfacilityTransportationCost_LD_M4; InterfacilityTransportationCost_LD_M5; InterfacilityTransportationCost_LD_M6;
Intermediate	HotA; ImpureE; IntAB; IntBC;
Inventory	-
InventoryByProduct	-
InventoryEconomicResources	-
InventoryEndProduct	InventoryEndProduct_1; InventoryEndProduct_1DC1; InventoryEndProduct_2; InventoryEndProduct_2DC1;
InventoryEnergeticResources	-
InventoryHoldingCost	-
InventoryHumanResources	-

InventoryIntermediate	InventoryIntermediate_AB; InventoryIntermediate_ABDC1;
InventoryMaterialResources	-
InventoryParameter	-
InventoryRawMaterial	Inv_A; Inv_B; Inv_C;
InventoryResidue	-
InventoryTransportResources	-
InventoryUnitResources	-
LocationCapacity	LocationCapacity_DC1; LocationCapacity_DC2; LocationCapacity_DC3; LocationCapacity_DC4; LocationCapacity_Supplier1; LocationCapacity_Supplier2; LocationCapacity_Supplier3;
LocationNumber	-
LocationParameter	-
Lot	-
Market	Market_1; Market_2; Market_3; Market_4; Market_5; Market_6;
MarketPlacement	LM1; LM2; LM3; LM4; LM5; LM6;
MasterProductionScheduling	-
MasterRecipe	MasterRecipe_129; MasterRecipe_209; MasterRecipe_253; MasterRecipe_333; MasterRecipe_5; MasterRecipe_85; MasterRecipe_P1; MasterRecipe_P2;
MaterialResourses	-
MixingSpeed	-
Mode	-
Operation	-
OtherAcquisitionCost	-
OutboundTransportationCost	-
P_Capacity	-
P_Materials	AB_P1; AB_P2_In; AB_P2_Out; A_P1; BC_P1; B_P1; C_P1; C_P2; E_P2; HA_P1; P1; P2; P_Materials_102; P_Materials_148; P_Materials_155; P_Materials_158; P_Materials_161; P_Materials_164; P_Materials_167; P_Materials_174; P_Materials_218; P_Materials_219; P_Materials_222; P_Materials_223; P_Materials_226; P_Materials_24; P_Materials_272; P_Materials_279; P_Materials_282; P_Materials_285; P_Materials_288; P_Materials_291; P_Materials_298; P_Materials_31; P_Materials_34; P_Materials_342; P_Materials_343; P_Materials_346; P_Materials_347; P_Materials_350; P_Materials_37; P_Materials_40; P_Materials_43; P_Materials_50; P_Materials_94; P_Materials_95; P_Materials_98; P_Materials_99;
P_MixingSpeed	-
P_Pressure	-
P_Temperature	-
P_Time	P_Time_107; P_Time_111; P_Time_185; P_Time_190; P_Time_196; P_Time_231; P_Time_235; P_Time_309; P_Time_314; P_Time_320; P_Time_355; P_Time_359; P_Time_61; P_Time_66; P_Time_72; P_Time_Heating_P1; P_Time_Reaction1_P1; P_Time_Reaction1_P2; P_Time_Reaction2_p1; P_Time_Separation_P2;
P_TransferRate	-
P_ValvePosition	-
Parameter	-
Phase	-
PhysicalModel	-
Pressure	-
ProceduralControl	-
ProceduralElement	-
ProceduralLink	Link_1; Link_2; Link_3; ProceduralLink_104; ProceduralLink_182; ProceduralLink_193; ProceduralLink_228; ProceduralLink_306; ProceduralLink_317; ProceduralLink_352; ProceduralLink_58; ProceduralLink_69;
ProceduralLogic	ProceduralLogic_103; ProceduralLogic_181; ProceduralLogic_227; ProceduralLogic_305; ProceduralLogic_351; ProceduralLogic_57; ProceduralLogic_P1; ProceduralLogic_P2;
Procedure	-
Process	-

ProcessAction	-
ProcessCell	ProcessCell_128; ProcessCell_252; ProcessCell_4; ProcessCell_kondili;
ProcessControl	-
ProcessInformation	-
ProcessingActivities	-
ProcessingTime	ProcessingTime_108; ProcessingTime_112; ProcessingTime_186; ProcessingTime_191; ProcessingTime_197; ProcessingTime_232; ProcessingTime_236; ProcessingTime_310; ProcessingTime_315; ProcessingTime_321; ProcessingTime_356; ProcessingTime_360; ProcessingTime_62; ProcessingTime_67; ProcessingTime_73; ProcessingTime_Heating; ProcessingTime_Reaction1; ProcessingTime_Reaction2; ProcessingTime_Reaction3; ProcessingTime_Separation;
ProcessManagement	-
ProcessOperation	Heating; ProcessOperation_113; ProcessOperation_187; ProcessOperation_198; ProcessOperation_237; ProcessOperation_311; ProcessOperation_322; ProcessOperation_361; ProcessOperation_63; ProcessOperation_74; Reaction; Separation;
ProcessOutput	-
ProcessParameter	-
ProcessStage	ProcessStage_106; ProcessStage_110; ProcessStage_184; ProcessStage_189; ProcessStage_195; ProcessStage_230; ProcessStage_234; ProcessStage_308; ProcessStage_313; ProcessStage_319; ProcessStage_354; ProcessStage_358; ProcessStage_60; ProcessStage_65; ProcessStage_71; ProcessStage_P1_Reaction1; ProcessStage_P1_Reaction2; ProcessStage_P1_heating; ProcessStage_P2_Reaction; ProcessStage_P2_Separation;
ProductionOrder	ProductionOrder_S1P1; ProductionOrder_S1P2;
ProductionParameter	-
ProductionPolicy	Earliness; EnvironmentalImpact; Makespan; Profit; Tardiness;
ProductionProcess	-
ProductionRequirement	-
ProductionScheduling	-
RawMaterial	FeedA; FeedB; FeedC;
RawMaterialCost	RawMaterialCostA-SP1; RawMaterialCostB-SP2; RawMaterialCostC-SP3;
RecipeElement	RecipeElement_118; RecipeElement_120; RecipeElement_203; RecipeElement_205; RecipeElement_207; RecipeElement_242; RecipeElement_244; RecipeElement_327; RecipeElement_329; RecipeElement_331; RecipeElement_366; RecipeElement_368; RecipeElement_79; RecipeElement_81; RecipeElement_83; RecipeElement_P11; RecipeElement_P12; RecipeElement_P13; RecipeElement_P21; RecipeElement_P22;
RecipeManagement	-
RecipeOperation	-
RecipePhase	-
RecipeProcedure	-
RecipeType	-
RecipeUnitProcedure	-
Requeriments	-
Residue	-
Resources	-
Restriction	-
SCInventoryManagement	SCInventoryManagement_DC;
SCLocationManagement	SCLocationManagement_DC1; SCLocationManagement_DC2; SCLocationManagement_DC3; SCLocationManagement_DC4; SCLocationManagement_Site1; SCLocationManagement_Site2; SCLocationManagement_Site3; SCLocationManagement_Site4; SCLocationManagement_Supplier1; SCLocationManagement_Supplier2; SCLocationManagement_Supplier3;
SCProductionandDistributionManagement	
SCTransportationManagement	-
Site	Site_1; Site_2; Site_3; Site_4;
SitePlacement	LA; LB; LC; LD;

SiteRecipe	SiteRecipe_123; SiteRecipe_247; SiteRecipe_371; SiteRecipe_kondili;
State	-
SuplierPlacement	GP_S1; GP_S2; GP_S3;
Supplier	Supplier_1; Supplier_2; Supplier_3;
SupplyChainCost	-
SupplyChainInformation	-
SupplyChainManagement	-
SupplyChainParameter	-
Temperature	-
Thing	-
Time	-
TranferRate	-
TransportationLink	TL_LA_LB; TL_LA_LC; TL_LA_LD; TL_LA_M1; TL_LA_M2; TL_LA_M3; TL_LA_M4; TL_LA_M5; TL_LA_M6; TL_LB_LC; TL_LB_LD; TL_LB_M1; TL_LB_M2; TL_LB_M3; TL_LB_M4; TL_LB_M5; TL_LB_M6; TL_LC_LD; TL_LC_M1; TL_LC_M2; TL_LC_M3; TL_LC_M4; TL_LC_M5; TL_LC_M6; TL_LD_M1; TL_LD_M2; TL_LD_M3; TL_LD_M4; TL_LD_M5; TL_LD_M6;
TransportationParameter	-
TransportedMaterials	TM_1; TM_2; TM_3; TM_4; TM_5; TM_6; TM_7; TM_8;
TransportResources	TransportResources_1; TransportResources_2;
Unit	HeatingReactor; Reactor1; Reactor2; SeparationUnit; Unit_10; Unit_131; Unit_134; Unit_140; Unit_16; Unit_211; Unit_255; Unit_258; Unit_264; Unit_335; Unit_7; Unit_87;
UnitProcedure	-
UnitResources	-
UnitSupervision	-
ValvePosition	-
WatingTime	-

6.4.2 Model remarks

This improvement in the model enhances the way for achieving a successful enterprise decision making supporting tool which adapts and recognizes the different elements found through the hierarchy models that are associated to the whole supply chain.

Moreover, a general semantic framework is presented, which is able to model any enterprise particular case, proving its re-usability. The ontology is used by applying it to an optimization framework as defined in chapter 5.

One of the benefits in using the ontology consists of the option of cloning a given site, or a part of it, within the model facilitates the definition of new sites within the supply chain structure, that can be the result of redesigning the supply chain. Therefore, the modeling of the supply chain may be achieved faster and with greater level of detail.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The work presented along the development of this thesis represents a step forward in the application of semantic models to the process industry. On the one hand, a methodology which comprises from the design and development of the ontological model to its final use has been proposed, applied and validated. On the other hand, the use of this tool in the process engineering domain has improved the decision making process along different decision levels. On the whole, the objectives posed initially have been achieved as explained next.

7.1.1 Methodology improvement

The field of ontology development has been thoroughly studied in order to design an adequate semantic model for representing the desired domain. As a result, the lack of a consensual methodology has been detected, along with the fact that the scope of the existing methodologies is limited to the methodological design, disregarding the application for the model exploitation in the domain. For this reason, in this thesis, a new methodology has been designed and reviewed through the application in different models, which considers the continuous improvement cycle. Such methodology allows an easy and friendly guide and reference on how to use the ontological model, and provides a guidance on the way the user can apply it as a decision tool.

Specifically, On-to-Knowledge and Methontology, which are two of the most commonly used existing methodologies, have been taken as reference for the development of the proposed methodology. Therefore, this work ensures an ordered quality manner to develop an ontology based on the well known continuous improvement tool PDSA cycle. From this point of view, this work stands for an ongoing effort to improve ontology process development in a friendly way, from the ontology design and building phase, to the practical framework application.

Although the results of applying this methodology have not been directly compared to the models that would have been obtained using other methodologies, it is clear that the wider scope of the proposed method has allowed to incorporate additional features to the domain, that would not have been obtained otherwise. For example, the final application of the semantic model, namely the decision support tools, has been considered from the design phase of the ontology, enabling higher success with the goals fulfillment.

On the whole, the proposed methodology has proved to have a large potential for the development and use of ontologies regardless the specific domain.

7.1.2 Semantic enterprise domain representation

The integration of various decision levels within a common representation facilitates the creation, storage and sharing of knowledge in a specific domain, and allows to improve the effectiveness of decision support systems.

In this thesis, the creation of a common model including the control, scheduling and supply chain hierarchical levels has been achieved. Thus, the gap between analytical techniques, such as optimization approaches, and transactional models concerned with the task processing and communicating data, has been reduced.

Furthermore, since the proposed enterprise ontology framework has been created using the standards, it has resulted in a well-behaved performance at capturing common understanding in conceptual design and at helping to utilize the relations that are mined from the databases. Such properties help users to obtain data and information in a proper, fast and standard way. Even more, it can be applied to web portals, facilitating the communication and knowledge sharing along different geographic locations.

The following list contains the main achievements related to the development of the enterprise ontology framework:

- A common and standardized language is established. As a result, better communication within the enterprise can be achieved, both from a formal perspective, using informatics languages, and an informal one, that is, in natural language among people within organizations. Therefore improved communication allows higher capacity to deal uncertainty, since it is easier to react against incidences which may stem from different sources and levels.

- This framework represents a modular and consistent procedure for standardizing processes at different decision levels, which helps in reducing engineering costs. Thus, it also be used as an enterprise management tool.

- The formal representation of the whole enterprise allows to check the consistency while instantiating of the model structure automatically, which results in a higher reliability of the decision systems that may be applied, and so of the final decisions made. Even more, the derived decision models, which can be either mathematical or coded for artificial intelligence purposes, can be automatically built.

- Based on a general ontological model comprising strategical (supply chain), tactical (scheduling) and operational (unit control) decision levels of the enterprise structure, a robust enough framework for enriching and assisting the decision making process along the aforementioned hierarchical levels has been created.

- Thus, by representing the whole system within a single model, coordination procedures can be improved allowing automation opportunities for manufacturing requirements.

- An integration bridge among the different enterprise processes and their related data has been developed. Therefore, by means of the exploitation of knowledge management and experience and by using different optimization tools, the generation and processing of quality information has been achieved. Thus, the information of the whole enterprise is available within the model, but only those pieces which are necessary are retrieved.

- In addition, this ontology opens the way for achieving successful flexible control in adapting and recognizing different elements found through the hierarchy models which are associated to manufacturing and processing at multilevel control systems.

- The limitations regarding the enterprise management capacity derived from the restriction to information access can be avoided by the use of the semantic model, which acts as an integrator in the decision making application. In addition, the proposed framework has the advantage of providing the data in any informatics format.

- The practical implementation of the ANSI/ISA standards to enterprise environments has been possible by the creation of a model according to the principles described in the standards. As a result, the instantiation of any particular problem within the model, agrees with the aforementioned standards.

- One of the most important achievements of the ontological model for the enterprise domain consists of its re-usability. Such property allows that this general semantic framework can model any enterprise particular case. Specifically, the instances of a problem can be derived straight forward based on the model features requirements.

- Another key achievement of the proposed framework is the usability, which lies on the fact that it can be applied to systematic tools for decision support related to different decision levels. In other words, the framework is usable to provide the adequate data to the decision maker in order to fulfill the enterprise goals.

- Furthermore, the model is represented in a formal language that can be (or become by automatic translation) a re-usable and/or shared component in software systems and hardware agents.

- Finally, this work represents a step forward to support the integration (not just "communication") of different software tools applicable to the management and exploitation of plant database information, resulting into an enhancement of the entire process management structure. Thus, since the ontology has been written in a shared language and can be opened by freeware software, it its a potential tool for any user.

7.2 Future work

Several aspects could be tackled in order to expand the scope of the enterprise ontology project, presented in this thesis. Specifically they can be grouped in the model domain and framework development issues. Such model improvements are described next:

- A user interface should be created in order to allow a user friendly framework, in accordance with the different particular decision hierarchical levels within the enterprise. In this way the final user should not have any background about ontology models or informatics programing in order to instantiated the desired reality (problem).

- The ontological model can be extended in the future to incorporate other functionalities of the enterprise domain, such as marketing issues, quality management, inventory management, proactive maintenance, strategical planning and so on. Thus, the incorporation within the model of production philosophies that are related to enterprises, for example the just in time, the six sigma, the 5 s´s, the e-business or the business intelligence may be tackled.

- Furthermore, the level of detail of the hierarchical structure can be even deeper. For example at process level, all the features of a given equipment (such as year of construction, material of construction, lifespan) which may be required for any decision task beyond the scope of these thesis.

- In order to maximize the usability of the model toward incorporating the decision making tools, their inclusion within the ontology, could be dealt. For example the mathematical models for optimization could be considered in the semantic model if the appropriate classes for mathematical language were created or imported.

- Finally another remarkable improvement that can be achieved to the presented ontological framework is the translation, with its adequate equivalence standard language, to other different geographical languages (such as Spanish, French, Chinese or German), in order to spread the usability among the geographically distributed partners of an enterprise.

Appendix A

Publications

This is a list of the works carried out so far within the scope of this thesis, in reversed chronological order.

A.1 Journals

A.1.1 Manuscripts published

Muñoz, E.; Capón-García, E.; Moreno-Benito, M.; Espuña, A.; Puigjaner, L. Scheduling and control decision-making under an integrated information environment. *Computers & Chemical Engineering*, ISSN: 0098-1354, 35 (5): 774 – 786 (**2011**).

Muñoz, E.; Espuña, A.; Puigjaner, L. Towards an ontological infrastructure for chemical batch process management. *Computers & Chemical Engineering*, ISSN: 0098-1354, 34(5): 668 – 682 (**2010**).

A.1.2 Manuscripts submitted

Muñoz, E.; Capón, E.; Espuña, A. Puigjaner, L. Ontological framework toward process system integration: Cases studies. *Computers and Chemical Engineering*, **2011**.

A.2 Conference proceeding articles

A.2.1 Articles in conference proceedings

Muñoz, E; Espuña, A.; Puigjaner, L. Integration of a multilevel control system in an ontological information environment. *European Symposium on Computer Aided Process Engineering* (ESCAPE-21), (J. Jeżowski and J. Thullie, Eds.), 648 – 652, ISBN: 978-0-444-53711-9, 2011.

Muñoz, E.; Botaro, G.; Espuña, A.; Puigjaner, L. Multilevel control system integration through an Ontological chemical flexible infrastructure. *American Institute of Chemical Engineers Meeting Conferencee Proceedings on CD-ROM* (AIChE Meeting-2010), (American Institute of Chemical Engineers), ISBN: 978-0-8169-1065-6, 2010.

Capón-García, E.; Moreno-Benito, M.; Muñoz, E.; Espuña, A.; Puigjaner, L. Scheduling and control decision-making under an integrated information environment. *European Symposium on Computer Aided Process Engineering* (ESCAPE-20), (S. Pieruzzi and G. Buzzi Ferraris, Eds.), 1195 – 1200, ISBN: 978-0-444-53569-6, 2010.

Muñoz, E.; Espuña, A.; Puigjaner, L. Ontological chemical flexible infrastructure light and heavy approach towards enterprise system integration. *American Institute of Chemical Engineers Meeting Conference Proceedings on CD-ROM* (AIChE Meeting-2009), (American Institute of Chemical Engineers), ISBN: 978-0-8169-1058-6, 2009.

Muñoz, E; Kopanos, G.M.; Espuña, A.; Puigjaner, L. Towards an ontological infrastructure for chemical batch process management. *European Symposium on Computer Aided Process Engineering* (ESCAPE-19), (J. Jeżowski and J. Thullie, Eds.), 883 – 888, ISBN: 978-0-444-53433-0, 2009.

A.2.2 Other congresses and workshops

Muñoz, E; Capon, E.; Laínez, J.M.; Espuña, A.; Puigjaner, L. Operational, Tactical and Strategic Integration for Enterprise Decision-Making. *European Symposium on Computer Aided Process Engineering* (ESCAPE-22), London, UK, 2012. (Accepted)

Silvente, J.; Muñoz, E; Espuña, A. Use of ontological structures for integrated Supply Chain Management. *European Symposium on Computer Aided Process Engineering* (ESCAPE-22), London, UK, 2012. (Accepted)

Muñoz, E.; Capón, E.; Espuña, A.; Puigjaner, L. Operational level integration system based on an ontological framework by means of master/control recipe semantic. *American Institute of Chemical Engineers Meeting Conferencee Proceedings on CD-ROM* (AIChE Meeting-2011), (American Institute of Chemical Engineers), Minneapolis, USA, 2011. (Oral presentation)

Muñoz, E.; Capón, E.; Lainez, J.M.; Espuña, A.; Puigjaner, L. Ontological framework for the enterprise from a process perspective. *International Joint Conference on Knowledge Engineering and Knowledge Management* (IC3K), Paris, France,2011. (Oral presentation)

Muñoz, E.; Capón, E.; Espuña, A.; Puigjaner, L. Integration of general master recipe modeling into process system ontology for improved automation. *12th Mediterranean Congress of Chemical Engineering*, Barcelona, Spain, 2011. (Oral communication)

Muñoz, E.; Kopanos, G.; Espuña, A.; Puigjaner, L. Batch Process Ontology. *11th Mediterranean Congress of Chemical Engineering*, Barcelona, Spain, 2008. (Poster presentation)

A.3 Participation in research projects

EHMAN, *Extendiendo los Horizontes productivos frente a la paradoja de la integración*, supported by the *Ministerio de Educación y Ciencia* (DPI2009-09386), Spain, 2010-2012.

ToleranT, *Sistema de Soporte Avanzado para Procesos de Fabricación Flexible en la Industria Química y Petroquímica*, supported by *Ministerio de Educación y Ciencia* (DPI2006-05673), Spain, 2007-2009.

Appendix B

Domain conceptualization

This appendix shows in Table B.1 the glossary of terms related to planning, scheduling and process control based on the ANSI/ISA-88 standard.

Table B.1: Concepts from ANSI/ISA-88

Name	Description
Administrator	A physical person capable of configuring the process coordinator.
Allocation	A form of coordination control that assigns a resource to a batch or unit. An allocation can be for the entire resource or for portions of a resource.
Arbitration	A form of coordination control that determines how a resource should be allocated when there are more requests for the resource than can be accommodated at one time.
Area	A component of a batch manufacturing site that is identified by physical, geographical, or logical segmentation within the site. An area may contain process cells, units, equipment modules, and control modules.
Basic control	Control that is dedicated to establishing and maintaining a specific state of equipment or process condition. Basic control may include regulatory control, interlocking, monitoring, exception handling, and discrete or sequential control.
Batch	The material that is being produced or that has been produced by a single execution of a batch process. An entity that represents the production of a material at any point in the process. Batch means both the material made by and during the process and also an entity that represents the production of that material. Batch is used as an abstract contraction of the words "the production of a batch."
Batch control	Control activities and control functions that provide a means to process finite quantities of input materials by subjecting them to an ordered set of processing activities over a finite period of time using one or more pieces of equipment.
Batch Process	A process that leads to the production of finite quantities of material by subjecting quantities of input materials to an ordered set of processing activities over a finite period of time using one or more pieces of equipment.
Batch production	A manufacturing process used to produce or process any product in batches.

Batch schedule	A list of batches to be produced in a specific process cell. The batch schedule typically contains such information as what is to be produced, how much is to be produced, when or in what order the batches are to be produced, and what equipment is to be used.
Common resource	A resource that can provide services to more than one requester. Common resources are identified as either exclusive-use resources or shared-use resources.
Control module	A Collection of sensors, actuators and associate processing equipment that acts as single entity from a control stand point, that can carry out basic control. This term applies to both the physical equipment and the equipment entity. A control module may contain another control modules.
Control recipe	A type of recipe which, through its execution, defines the manufacture of a single batch of a specific product.
Coordination control	A type of control that directs, initiates, and/or modifies the execution of procedural control and the utilization of equipment entities.
Enterprise	An organization that coordinates the operation of one or more sites. Must contain a site.
Equipment control	The equipment-specific functionality that provides the actual control capability for an equipment entity, including procedural, basic, and coordination control, and that is not part of the recipe.
Equipment entity	A collection of physical processing and control equipment and equipment control grouped together to perform a certain control function or set of control functions.
Equipment module	A functional group of equipment that can carry out a finite number of specific minor processing activities. An equipment module is typically centered around a piece of process equipment (a weigh tank, a process heater, a scrubber, etc.). This term applies to both the physical equipment and the equipment entity. An equipment module may contain another equipment modules, and control modules.
Equipment operation	An operation that is part of equipment control. An equipment operation has an ordered set of equipment phase
Equipment phase	A phase that is part of equipment control.
Equipment procedure	A procedure that is part of equipment control. An equipment procedure has an ordered set of equipment unit procedure.
Equipment unit procedure	A unit procedure that is part of equipment control. An equipment unit procedure has an ordered set of equipment phase.
Exception handling	Those functions that deal with plant or process contingencies and other events which occur outside the normal or desired behavior of batch control.
Exclusive-use resource	A common resource that only one user can use at any given time.
Failures diagnosis tool	A system that in in charge of monitor the state of the plant for detect and identify anomalous situations, generates alarms and propose a corrective actions system.
Formula	A category of recipe information that includes process inputs, process parameters, and process outputs.
General recipe	A type of recipe that expresses equipment and site independent processing requirements.
Header	Information about the purpose, source and version of the recipe such as recipe and product identification, creator, and issue date.
ID	A unique identifier for batches, lots, operators, technicians, and raw materials.
Lot	A unique amount of material having a set of common traits.
Master recipe	A type of recipe that accounts for equipment capabilities and may include process cell-specific information.
Mode	The manner in which the transition of sequential functions are carried out within a procedural element or the accessibility for manipulating the states of equipment entities manually or by other types of control.
Operation	A procedural element defining an independent processing activity consisting of the algorithm necessary for the initiation, organization, and control of phases. An operation has an ordered set of phases.
Path; stream	The order of equipment within a process cell that is used, or is expected to be used, in the production of a specific batch.

Phase	The lowest level of procedural element in the procedural control model.
Procedural control	Control that directs equipment-oriented actions to take place in an ordered sequence in order to carry out some process-oriented task. Has a building block of procedural element
Procedural element	A building block for procedural control that is defined by the procedural control model.
Procedure	The strategy for carrying out a process. In general, it refers to the strategy for making a batch within a process cell. It may also refer to a process that does not result in the production of product, such as a clean-in-place procedure. A procedure has an ordered set of unit procedure.
Process	A sequence of chemical, physical, or biological activities for the conversion, transport, or storage of material or energy. A process consists of an ordered set of Process Stage.
Process Action	Minor processing activities that are combined to make up a process operation.
Process cell	A logical grouping of equipment that includes the equipment required for production of one or more batches. It defines the span of logical control of one set of process equipment within an area. This term applies to both the physical equipment and the equipment entity. Must contain unit.
Process control	The control activity that includes the control functions needed to provide sequential, regulatory, and discrete control and to gather and display data.
Process Coordinator	A system component that is in charge of process management, unit supervision and process control. If an abnormal situation is received it executes a restore routine (if it is required).
Process input	The identification and quantity of a raw material or other resource required to make a product.
Process management	The control activity that includes the control functions needed to manage batch production within a process cell.
Process Operation	A major processing activity that usually results in a chemical or physical change in the material being processed and that is defined without consideration of the actual target equipment configuration. A process operation consists of an ordered set of process actions.
Process output	An identification and quantity of material or energy expected to result from one execution of a control recipe.
Process parameter	Information that is needed to manufacture a material but does not fall into the classification of process input or process output.
Process Stage	A part of a process that usually operates independently from other process stages and that usually results in a planned sequence of chemical or physical changes in the material being processed. A process stage consists of an ordered set of process operations.
Production information management	The activity that is in charge of the compilation, processing and notification of the information of production.
Production planning and scheduling	The activity that is formed by decision algorithms for producing plan production.
Production Process	The production process is concerned with transforming a range of inputs process into those outputs that are required by the market by the use of transforming resources .There are three main types of production process: job, batch and flow production.
Recipe	The necessary set of information that uniquely defines the production requirements for a specific product. There are four types of recipes defined in S88.01: general, site, master, and control. A recipe may contain a header, a formula, an equipment, requirements, a procedure and other information.
Recipe management	The control activity that includes the control functions needed to create, store, and maintain general, site, and master recipes.
Recipe operation	An operation that is part of a recipe procedure in a master or control recipe. Recipe operation is an ordered set of recipe phase.
Recipe phase	A phase that is part of a recipe procedure in a master or control recipe.
Recipe procedure	The part of a recipe that defines the strategy for producing a batch. A recipe procedure is an ordered set of unit procedure, recipe operation and recipe phase.

Recipe unit procedure	A unit procedure that is part of a recipe procedure in a master or control recipe. Recipe procedure is an ordered set of recipe operation.
Requirement	A singular documented need of what a particular product or service should be or do.
Shared-use resource	A common resource that can be used by more than one user at a time.
Site	A component of a batch manufacturing enterprise that is identified by physical, geographical, or logical segmentation within the enterprise. A site may contain areas, process cells, units, equipment modules, and control modules.
Site recipe	A type of recipe that is site specific. Site recipes may be derived from general recipes recognizing local constraints, such as language and available raw materials.
State	The condition of an equipment entity or of a procedural element at a given time. The number of possible states and their names vary for equipment and for procedural elements.
System Components	All the components needed for manage and control a batch production system.
Train; line	A collection of one or more units and associated lower level equipment groupings that has the ability to be used to make a batch of material.
Unit	A collection of associated control modules and/or equipment modules and other process equipment in which one or more major processing activities can be conducted. Units are presumed to operate on only one batch at a time. Units operate relatively independently of one another. This term applies to both the physical equipment and the equipment entity. A unit may contain equipment modules, and control modules.
Unit procedure	A strategy for carrying out a contiguous process within a unit. It consists of contiguous operations and the algorithm necessary for the initiation, organization, and control of those operations. Unit procedure has an ordered set of operations.
Unit recipe	The part of a control recipe that uniquely defines the contiguous production requirements for a unit. The unit recipe contains the unit procedure and its related formula, header, equipment requirements, and other information.
Unit supervision	The control activity that includes control functions needed to supervise the unit and the unit's resources.

Appendix C

Interrelationship matrix

This appendix presents the interrelation matrix of the process description based on ANSI/ISA standards.

	Batch Process	Batch	Recipe	Header	Formula	Process	Process Stage	Process Operation	Process Action	Enterprise	Site	Area	Process cell	Unit	Equipment module	Control module	Procedure	Unit procedure	Operation	Phase	Basic control	Coordination control	Allocation	Arbitration	Recipe management	General recipe	Site recipe	Master recipe	Control recipe	Batch control	Batch schedule	Common resource	Equipment control	Equipment entity	Equipment procedure	Equipment unit procedure	Equipment operation	Equipment phase	Exception handling	Exclusive-use resource	ID	Lot	Mode	Path; stream	Procedural control	Procedural element	Process control	Process input	Process output	Process management	Process parameter	Recipe procedure	Recipe unit procedure	Recipe operation	Recipe phase	Shared-use resource	State	Train; line	Unit recipe	Unit supervision	Production planning and scheduling	Production information management	Failures diagnosis tool	Requirement	Production Process System	Batch production	Transforming resourses	Process Coordinator	System Components	Administrator	Scheduling system (MOPP)	Pilot plant (PROCEL)	Data Manager (Thot)	Data Base (ISA S88)	Failures diagnosis tool (ChemEye)	Other
Batch Process		has Batch	has recipe element																																													has process input	has process output	has process management																										
Batch	is part of																							recive information from																																																						
Recipe	is part of			receive information from																																																																								
Header			provide information to		receive information from																																																																							
Formula			provide information to																																																																								
Process																																																																												
Process Stage						is order set of		has order set of																																																																				
Process Operation							is order set of		has order set of																																																																			
Process Action								is ortder set of																																																																				
Enterprise											has site																																																																		
Site										is site of		has area																																																																
Area											is part of						has procedure																																																											
Process cell														has unit																																																													
Unit													is part of		has equipment module									recive information from																																																							
Equipment module			provide information to											is part of	has equipment module	has control module																																																												
Control module															is part of	has control module																	is part of																																												
Procedure																		has order set of																																																											
Unit procedure																	is order set of		has order set of																																																										
Operation																		is order set of		has order set of																																																								
Phase																			is order set of																																																									
Basic control																																	receive information from																																												
Coordination control																						has Allocation	has arbitration									receive information from																																												
Allocation		provide information to									provide information to																																																																		
Arbitration		provide information to									provide information to																																																																		
Recipe management																									include	include	include	include																																																				
General recipe																								derives from / include in		derives in																																																						
Site recipe																								include in	derives from		derives in																																																				
Master recipe																								include in		derives from		derives in																																																			
Control recipe																								include in																																																								
Batch control																																	has equipment control														is part of																													
Batch schedule																																																																												
Common resource																																																																												
Equipment control																has equipment part					provide information to	provide information to								is part of					has equipment part	has equipment part	has equipment part												provide information to																															
Equipment entity													has order set of	has order set of	has order set of																																										has some																			
Equipment procedure																																	is part of			has order set of	has order set of																																											
Equipment unit procedure																																	is part of		is order set of																																														
Equipment operation																																	is part of		is order set of			has order set of																																										
Equipment phase																																				is order set of																																												
Exception handling																																																															derives from													
Exclusive-use resource																																																																												
ID			provide information to																																																																									
Lot																																																																												
Mode																																																																												has manner
Path; stream																																																																												has equipment order
Procedural control																																														has P-E building bloks																														
Procedural element																																	receive information from												is a bulding block for P-C												has some																			
Process control																														has Batch Control																																																			
Process input	is part of																																																																											
Process output	is part of																																																																											
Process management	is part of																																																																											
Process parameter																																																																												has manufacturung information
Recipe procedure																																																					has order set of																							has lower level unit
Recipe unit procedure																																																				is order set of		has order set of																						
Recipe operation																																																					is order set of		has order set of																					
Recipe phase																																																						is order set of																						
Shared-use resource																																																																												
State																																		is part of												is part of																														has state condition
Train; line														has some																																																														is asociate a lower level
Unit recipe																																																																												
Unit supervision																																																																												
Production planning and scheduling																																																																												
Production information management																																																																												
Failures diagnosis tool																																							derives in																																												
Requirement			provide information to																																																																									
Production Process System																																																																					has production process			has some							
Batch production																																																																				is type of											
Process Coordinator																																															is in charge of process control			is in charge of process management										is in change of unit supervision										=						
System Components																																																																				is part of											
Administrator																																																																							=								
Other																																																																												manner; equipment order; manufacturing information; state condition; lower level unit

Appendix D

Java code

This appendix contains the Java code for the application of the enterprise ontology project.

```


/* To change this template, choose Tools | Templates and open the template in the editor. */

package ontologia3;
import java.io.*;
import java.io.File;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLClassExpression;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLDataPropertyExpression;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLLiteral;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectPropertyExpression;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.UnloadableImportException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import org.semanticweb.owlapi.model.AxiomType;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLLogicalAxiom;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLSubClassOfAxiom;

/* @author Edrisi */

public class Main {

    private static OWLSubClassOfAxiom SubTypeOf;
/* @param args the command line arguments */
   public static void writetxt(String name, String instancias) {
    try {
           FileWriter outFile = new FileWriter(name);
           PrintWriter out = new PrintWriter(outFile);

           // Write text to file
           String[] ins = instancias.split(" ");
           for(int j = 0; j < ins.length ; j++){
                out.println(ins[j]);
           }
           out.close();
      } catch (IOException e){
            e.printStackTrace();
      }
   }



   public static void write2txt(String name, String instancias){
    try {
           FileWriter outFile = new FileWriter(name);
           PrintWriter out = new PrintWriter(outFile);

           // Write text to file
           String[] ins = instancias.split("&");
           for(int j = 0; j < ins.length ; j++){
                out.println(ins[j]);
           }
           out.close();
      } catch (IOException e){
            e.printStackTrace();
      }
   }


      public static String findmyindividuals(String nameIndividual, String property, OWLOntology bapron, OWLOntologyManager manager){
//  funcion que busca las intancias que se encuentran instanciadas dentro de una propiedad especifica de una instancia name//
//  los datos de entrada son
          //1. el nombre de la instancia a ser analizada
          //2. el nombre de la propiedad de la cual se desean obtener las instancias
          //3. el nombre de la ontologia
          //4. el administrador del archivo de la ontologia

// funcion que encuentra el IRI de la instancia.
       IRI myindividualID = IRI.create(bapron.getOntologyID().getOntologyIRI() + "#" +nameIndividual);

//obtiene el nombre de la instancia relacionado con ese IRI.
       OWLIndividual myindividual = manager.getOWLDataFactory().getOWLNamedIndividual(myindividualID);

//obtenemos las propiedades de la instacia de receta maestra//
         Map<OWLObjectPropertyExpression, Set<OWLIndividual>> relements = myindividual.getObjectPropertyValues(bapron);

//guardamos un string de las propiedades con la variable var1 //
         String var1 = relements.toString();

//separamos las diferentes propiedades y sus instancias //
         String[] var2 = var1.split(">], <");
         int Dove = 0;
         for (int j = 0; j < var2.length; j++){
             String[] var21 = var2[j].split(">=");
             String[] var22 = var21[0].split("#");
             if (var22 [1].equals(property))
                Dove = j;
         }

 //separa la propiedad deseada "hasrecipeelement" y sus instancias "RE001 etc"
```

```
            String[] var3 = var2[Dove].split("=");

//separamos las instancias en lineas"
            String[] var4 = var3[1].split(",");

//creamos el string donde se guardan las instancias en cada loop que hace//
             String var5 = "";
             for(int j = 0; j < var4.length; j++){

//eliminamos el IRI de la instancia//
                   String[] auxvar = var4[j].split("#");

//eliminamos el > tambien de la instancia//
                    String[] auxvar2 = auxvar[1].split(">");
                       var5 = var5 + auxvar2[0] + " ";
                 }
             return var5;
    }


        public static String findmyvalues(String nameIndividual, String property, OWLOntology bapron, OWLOntologyManager manager){

//funcion que busca las intancias que se encuentran instanciadas dentro de una propiedad especifica de una instancia name//
//los datos de entrada son
             //1. el nombre de la instancia a ser analizada
             //2. el nombre de la propiedad de la cual se desean obtener las instancias
             //3. el nombre de la ontologia
             //4. el administrador del archivo de la ontologia

// funcion que encuentra el IRI de la instancia.
           IRI myindividualID = IRI.create(bapron.getOntologyID().getOntologyIRI() + "#" +nameIndividual);
//obtiene el nombre de la instancia relacionado con ese IRI.
           OWLIndividual myindividual = manager.getOWLDataFactory().getOWLNamedIndividual(myindividualID);

 //obtenemos las propiedades de la instacia de receta maestra//
            Map<OWLDataPropertyExpression,java.util.Set<OWLLiteral>> relements = myindividual.getDataPropertyValues(bapron);

```

```
//guardamos un string de las propiedades con la variable var1 //
          String var1 = relements.toString();

//separamos las diferentes propiedades y sus instancias //
          String[] var2 = var1.split(",");
          int Dove = 0;
          for (int j = 0; j < var2.length; j++){
              String[] var21 = var2[j].split(">=");
              String[] var22 = var21[0].split("#");
              if (var22 [1].equals(property))
                 Dove = j;
          }
            /*String aux9Wkt = "";
                             String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron, manager);
                             String [] auxWkt = Wkt.split(" ");
                             *** IRI myqoutID = IRI.create(bapron.getOntologyID().getOntologyIRI() + "#" +aux5[0]);
                             ***OWLIndividual myqout = manager.getOWLDataFactory().getOWLNamedIndividual(myqoutID);
                             ***** MAP!!!!
                             ****String qout = qoutvalue.toString();
                             String[] qout1 = qout.split(",");
                             aux9Wkt = aux9Wkt + auxWkt[0] + "       " + qout1[1] +"&";
                             GenWkt = GenWkt + aux9Wkt;
                             write2txt("Workspace\\JAVA\\Ontologia3\\Results\\StructuredOutput\\" + "CEnEdProduct" +".txt", GenWkt);
                             }*/
          //separa la propiedad deseada "hasrecipeelement" y sus instancias "RE001 etc"
          String[] var3 = var2[Dove].split("\"");
//separamos las instancias en lineas"
          String var4 = var3[1];

          return var4;
     }

   public static float hoursBetween2Dates(Calendar c1, Calendar c2) {
```

```
                return ((c2.getTime().getTime() - c1.getTime().getTime())/ (3600 * 1000));
        }


    public static void main(String[] args) {

        try{
            String micarpeta = "Workspace\\JAVA\\ScheduleResults\\";
            String micarpetaSC = "Workspace\\JAVA\\SCResults\\";

            OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
//              File file = new File("Workspace\\JAVA\\ejemplo1_Bapron.owl");
//              File file = new File("Workspace \\JAVA\\ejemplo2_Kondili.owl");
//              File file = new File("Workspace \\ejemplo3_Integration.owl");
                File file = new File("Workspace \\ejemplo4_KondiliSC.owl");


            OWLOntology bapron = manager.loadOntologyFromOntologyDocument(file);


            System.out.print("INICIO BET" + "\n");

            System.out.println("Loaded ontology: " + bapron);
            Set<OWLClass> clases2 = bapron.getClassesInSignature();
            Iterator itclas = clases2.iterator();


            OWLSubClassOfAxiom jmn = SubTypeOf;
            Set<OWLSubClassOfAxiom>  mln = bapron.getAxioms(jmn);


//Obtain data and object properties.

            Set<OWLDataProperty> datap = bapron.getDataPropertiesInSignature();
            Iterator itdprop = datap.iterator();

```

```
Set<OWLObjectProperty> objectp = bapron.getObjectPropertiesInSignature();
Iterator itoprop = objectp.iterator();

String DProperties = " ";
String DDProperties = " ";
String RDProperties = " ";

while (itdprop.hasNext()){
    OWLDataProperty k = (OWLDataProperty) itdprop.next();
    String g = k.getIRI().toString();
    String[] aux22 = g.split("#");
    DProperties = DProperties + aux22[1] + "  ";
}


String OProperties = " ";
String DOProperties = " ";
String ROProperties = " ";
while (itoprop.hasNext()){
    OWLObjectProperty k = (OWLObjectProperty) itoprop.next();
    String g = k.getIRI().toString();
    String[] aux22 = g.split("#");

    Set<OWLClassExpression> dd = k.getDomains(bapron);

    Iterator itcprop = dd.iterator();

    String COproperties = "";

    while (itcprop.hasNext()){
    OWLClassExpression ebet = (OWLClassExpression) itcprop.next();
    Set<OWLClass> dd2 = ebet.getClassesInSignature();

    Iterator itcprop2 = dd2.iterator();

    while (itcprop2.hasNext()){
         OWLClassExpression ebet2 = (OWLClassExpression) itcprop2.next();
         OWLClass nameedbet2 = ebet2.asOWLClass();

        String h = nameedbet2.getIRI().toString();
        String[] auxlks = h.split("#");
```

```
                    COproperties = COproperties + auxlks[1]+ ";";
                }

                if (COproperties.equals(""))
                        COproperties="none";
                Set<OWLClassExpression> dd1 = k.getRanges(bapron);
                Iterator itcprop1 = dd1.iterator();
                String COproperties1 = "";

                while (itcprop1.hasNext()){
                OWLClassExpression ebet1 = (OWLClassExpression) itcprop1.next();
                Set<OWLClass> dd21 = ebet1.getClassesInSignature();

                Iterator itcprop21 = dd21.iterator();

                while (itcprop21.hasNext()){
                     OWLClassExpression ebet21 = (OWLClassExpression) itcprop21.next();
                     OWLClass nameedbet21 = ebet21.asOWLClass();

                    String h = nameedbet21.getIRI().toString();
                    String[] auxlks1 = h.split("#");
                    COproperties1 = COproperties1 + auxlks1[1]+ ";";
                }
                }
                OProperties = OProperties + aux22[1] + "\t" + COproperties + "\t" + COproperties1 + "\n";
            }
            }
            writetxt(micarpeta + "Output\\" + "object properties" + ".txt", OProperties);

//  End obtain data and object Properties


// Obtain data for the Scheduling and SC optimization frameworks

            while(itclas.hasNext()){
                OWLClass c = (OWLClass) itclas.next();
                String s = c.getIRI().toString();
```

```
String[] aux3 = s.split("#");

// Condicion de clase Master Recipe: Rtn.
Boolean Rtn=false;
    if (aux3 [1].equals("MasterRecipe"))
       Rtn = true;

// Condition of class InventoryMaterialResources: Bh.
    Boolean Bh=false;
    if (aux3 [1].equals("InventoryMaterialResources"))
       Bh = true;

  // Condition of class InventoryMaterialResources: Bh.
    Boolean Bh1=false;
    if (aux3 [1].equals("ByProduct"))
     Bh1 = true;

// Condition of class InventoryMaterialResources: Bh.
    Boolean Bh2=false;
    if (aux3 [1].equals("EndProduct"))
       Bh2 = true;

// Condition of class InventoryMaterialResources: Bh.
    Boolean Bh3=false;
    if (aux3 [1].equals("Intermediate"))
       Bh3 = true;

 // Condition of class InventoryMaterialResources: Bh.
    Boolean Bh4=false;
    if (aux3 [1].equals("Residue"))
       Bh4 = true;

    // Condition of class InventoryMaterialResources: Bh.
    Boolean Vc=false;
    if (aux3 [1].equals("EndProduct"))
       Vc = true;

    // Condition of class Site: Pll1.
    Boolean Pll1=false;
    if (aux3 [1].equals("Site"))
       Pll1 = true;
```

```

                // Condition of class TransportResources: Pll2.
                Boolean Pll2=false;
                if (aux3 [1].equals("TransportResources"))
                  Pll2 = true;

                 // Condition of class TransportResources: Pch1.
                Boolean Pch1=false;
                if (aux3 [1].equals("TransportedMaterials"))
                  Pch1 = true;

                // Condition of class FacilityInvestmentCos: Pch2.
                Boolean Pch2=false;
                if (aux3 [1].equals("FacilityInvestmentCost"))
                  Pch2 = true;

                // Condition of class Market: Spt.
                Boolean Spt=false;
                if (aux3 [1].equals("Market"))
                  Spt = true;

                // Condition of class RawMaterial: Spt2.
                Boolean Spt2=false;
                if (aux3 [1].equals("RawMaterialCost"))
                  Spt2 = true;

                // Condition of class LocationCapacity: FrE.
                Boolean FrE=false;
                if (aux3 [1].equals("LocationCapacity"))
                  FrE = true;

                // Condition of class TL: FrE2.
                Boolean FrE2=false;
                if (aux3 [1].equals("TransportationLink"))
                  FrE2 = true;

```

```
             // Condition of class SitePlacement: Ptrr.
             Boolean Ptrr=false;
             if (aux3 [1].equals("SitePlacement"))
                Ptrr = true;

             // Condition of class MarketPlacement: Ptrr2.
             Boolean Ptrr2=false;
             if (aux3 [1].equals("MarketPlacement"))
                Ptrr2 = true;


        // Create Strings to be used in the output files.
             String GenWkt = "";
             String GenWkt1 = "";
             String GenWkt2 = "";
             String GenWkt3 = "";
             String GenWkt4 = "";
             String GenWkt5 = "";
             String GenWkt6 = "";
             String GenWkt7 = "";
             String GenWkt8 = "";
             String GenWkt11 = "";

             String DemandedProducts = "";
             Boolean findpll1 = true;
             String auxmpc = "";
             String auxmpc2 = "";
             String auxmpc3 = "";
             String Maxcapacity = "";
             String auxkpon = "";
              // End create Strings to be used in the output files.


             if(aux3.length > 1)
                 System.out.println(aux3[1]);
         Set<OWLIndividual> aux = c.getIndividuals(bapron);
         Iterator itind = aux.iterator();
             String aux6 = "";
             int i=0;
             while(itind.hasNext()){
                 OWLIndividual p = (OWLIndividual) itind.next();
```

```
                              String aux1 = p.toString();
                              String[] aux4 = aux1.split("#");
                              String[] aux5 = aux4[1].split(">");
                              aux6  = aux6 + aux5[0] + " ";
                              System.out.print(aux5[0] + " ");
                              i++;


//**********************************************************************
// Part belonging to the SCHEDULING //**********************************************************************

//Rtn es la condicion de la Master recipe: Busca los ids del process stage que pertenecen a la master recipe//
                             if (Rtn) {
                             String myname = findmyindividuals(aux5[0],"hasID_RecipeID",bapron,manager);
                             String Wkt = findmyindividuals(aux5[0],"hasRecipeElement",bapron,manager);
                             String [] auxWkt = Wkt.split(" ");
                             String aux2Wkt = "";
                             String aux3Wkt = "";
                             String aux4Wkt = "";
                             String aux5Wkt = "";
                             String aux7Wkt = "";
                             String aux8Wkt = "";
                             String aux9Wkt = "";
                             for(int j = 0; j < auxWkt.length; j++){
                             String Gorr = findmyindividuals(auxWkt[j],"hasIDFromRecipeElementType",bapron,manager);
                             String[] Gorr11 = Gorr.split(" ");
                             String Gorr2 = findmyindividuals(auxWkt[j],"hasEquipmentRequirement",bapron,manager);
                             String[] Gorr22 = Gorr2.split(" ");
                             String Gorr3 = findmyindividuals(Gorr22[0],"hasID_EquipmentRequirement",bapron,manager);
```

```
                            String Gorr4 = findmyindividuals(
                              Gorr11[0],"
                              receiveInformationFromIDRecipeElementType
                              ",bapron,manager);
                            String[] Gorr44 = Gorr4.split(" ");
                            String Gorr5 = findmyindividuals(
                              Gorr44[0],"
                              hasProcessInputParameter",bapron,
                              manager);
                            String[] Gorr55 = Gorr5.split(" ");
                            String Gorr6 = findmyindividuals(
                              Gorr44[0],"
                              hasProcessOutputParameter",bapron
                              ,manager);
                            String[] Gorr66 = Gorr6.split(" ");
                            String Gorr7 = findmyindividuals(
                              Gorr44[0],"hasTime",bapron,
                              manager);
                            String[] Gorr77 = Gorr7.split(" ");

                             // Find the IRI of this instance.
                            IRI mytimeID = IRI.create(bapron.
                              getOntologyID().getOntologyIRI()
                              + "#" +Gorr77[0]);

//Obtain the name of the instance with this  IRI.

                            OWLIndividual mytime = manager.
                              getOWLDataFactory().
                              getOWLNamedIndividual(mytimeID);
//obtenemos las propiedades de la instacia de receta maestra//
                            Map<OWLDataPropertyExpression,java.
                              util.Set<OWLLiteral>> timevalue =
                               mytime.getDataPropertyValues(
                              bapron);
                            String Ptime = timevalue.toString()
                              ;
                            String[] Ptime2 = Ptime.split("\"")
                              ;
                            aux7Wkt = aux7Wkt + myname + "." +
                              Gorr + "." + Gorr3 + "         " +
                              Ptime2[1] + "&";

////eliminamos el IRI de la instancia//
                            aux2Wkt = aux2Wkt + myname + "." +
                              Gorr + "&";
                            aux3Wkt = aux3Wkt + myname + "." +
                              Gorr + "." + Gorr3 + "&";
```

```
                                for(int k = 0; k < Gorr55.length; k
                                  ++){
                                String Gorr551 = findmyindividuals(
                                  Gorr55[k],"hasParameterSource",
                                  bapron,manager);
                                String[] Gorr552 = Gorr551.split("
                                  ");
                                String Gorr5521 = findmyindividuals
                                  (Gorr552[0],"hasID_Material",
                                  bapron,manager);
                                String[] Gorr5522 = Gorr5521.split
                                  (" ");
                                aux4Wkt = aux4Wkt + myname + "." +
                                  Gorr + "." + Gorr5522[0] + "&";

                                IRI myqinID = IRI.create(bapron.
                                  getOntologyID().getOntologyIRI()
                                  + "#" +Gorr55[k]);
                                OWLIndividual myqin = manager.
                                  getOWLDataFactory().
                                  getOWLNamedIndividual(myqinID);
                                Map<OWLDataPropertyExpression,java.
                                  util.Set<OWLLiteral>> qinvalue =
                                  myqin.getDataPropertyValues(
                                  bapron);
                                String qin = qinvalue.toString();
                                String[] qin1 = qin.split("\"");
                                aux8Wkt = aux8Wkt + myname + "." +
                                  Gorr + "." + Gorr5522[0] + "
                                        " + qin1[1]   + "&";
                                }
                                for(int k = 0; k < Gorr66.length; k
                                  ++){
                                String Gorr551 = findmyindividuals(
                                  Gorr66[k],"hasParameterSource",
                                  bapron,manager);
                                String[] Gorr552 = Gorr551.split("
                                  ");
                                String Gorr5521 = findmyindividuals
                                  (Gorr552[0],"hasID_Material",
                                  bapron,manager);
                                String[] Gorr5523 = Gorr5521.split
                                  (" ");
                                aux5Wkt = aux5Wkt + myname + "." +
                                  Gorr + "." + Gorr5523[0] + "&";

                                IRI myqoutID = IRI.create(bapron.
                                  getOntologyID().getOntologyIRI()
```

```
            + "#" +Gorr66[k]);
          OWLIndividual myqout = manager.getOWLDataFactory().getOWLNamedIndividual(myqoutID);
          Map<OWLDataPropertyExpression,java.util.Set<OWLLiteral>> qoutvalue = myqout.getDataPropertyValues(bapron);
          String qout = qoutvalue.toString();
          String[] qout1 = qout.split("\"");
          aux9Wkt = aux9Wkt + myname + "." + Gorr + "." + Gorr5523[0] + " " + qout1[1] +"&";
          }
          }
          GenWkt = GenWkt + aux2Wkt;
          GenWkt1 = GenWkt1 + aux3Wkt;
          GenWkt2 = GenWkt2 + aux4Wkt;
          GenWkt3 = GenWkt3 + aux5Wkt;
          GenWkt4 = GenWkt4 + aux7Wkt;
          GenWkt7 = GenWkt7 + aux8Wkt;
          GenWkt8 = GenWkt8 + aux9Wkt;

          System.out.println(GenWkt);
          write2txt(micarpeta + "StructuredOutput\\" + "sprocess" +".set", GenWkt);
          write2txt(micarpeta + "StructuredOutput\\"  + "prodstageunit" +".set", GenWkt1);
          write2txt(micarpeta + "StructuredOutput\\" + "psin" +".set", GenWkt2);
          write2txt(micarpeta + "StructuredOutput\\"  + "psout" +".set", GenWkt3);
          write2txt(micarpeta + "StructuredOutput\\" + "pt" +".txt", GenWkt4);
          write2txt(micarpeta + "StructuredOutput\\" + "qpsin" +".txt", GenWkt7);
          write2txt(micarpeta + "StructuredOutput\\" + "qpsout" +".txt", GenWkt8);
          }
        if (Bh) {
          String aux6Wkt = "";
```

```
                String myname = findmyindividuals(aux5[0],"hasResource",bapron,manager);
                String [] myname2 = myname.split(" ");
                String Wkt = findmyindividuals(myname2[0],"hasID_Material",bapron,manager);
                String [] auxWkt = Wkt.split(" ");
                String aux2Wkt = "";
                Map<OWLDataPropertyExpression,java.util.Set<OWLLiteral>> inventoryvalue = p.getDataPropertyValues(bapron);
                String inventory = inventoryvalue.toString();
                String[] inventory2 = inventory.split("\"");
                aux6Wkt = aux6Wkt + auxWkt[0] + " " + inventory2[1] + "&";

                GenWkt = GenWkt + aux6Wkt;
                write2txt(micarpeta + "StructuredOutput\\" + "capacity" +".txt", GenWkt);
                }

            if (Bh1) {
                String aux9Wkt = "";
                String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron,manager);
                String [] auxWkt = Wkt.split(" ");
                 String Wktval = findmyvalues(aux5[0],"max_value",bapron,manager);
                aux9Wkt = aux9Wkt + auxWkt[0] + " " + Wktval +"&";
                GenWkt = GenWkt + aux9Wkt;
                write2txt(micarpeta + "StructuredOutput\\" + "CByProduct" +".txt", GenWkt);
                }

            if (Bh2) {
                   String aux9Wkt = "";
                String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron,manager);
```

```
      String [] auxWkt = Wkt.split(" ");
      String Wktval = findmyvalues(aux5[0],"max_value",bapron,manager);
      aux9Wkt = aux9Wkt + auxWkt[0] + " " + Wktval +"&";
      GenWkt = GenWkt + aux9Wkt;
      write2txt(micarpeta + "StructuredOutput\\" + "CEndProduct" +".txt", GenWkt);
    }

    if (Bh3) {
      String aux9Wkt = "";
      String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron,manager);
      String [] auxWkt = Wkt.split(" ");
       String Wktval = findmyvalues(aux5[0],"max_value",bapron,manager);
      aux9Wkt = aux9Wkt + auxWkt[0] + " " + Wktval +"&";
     GenWkt = GenWkt + aux9Wkt;
      write2txt(micarpeta + "StructuredOutput\\" + "CIntermediate" +".txt", GenWkt);
      }

    if (Bh4) {
      String aux9Wkt = "";
      String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron,manager);
      String [] auxWkt = Wkt.split(" ");
     String Wktval = findmyvalues(aux5[0],"max_value",bapron,manager);
      aux9Wkt = aux9Wkt + auxWkt[0] + " " + Wktval +"&";
      GenWkt = GenWkt + aux9Wkt;
      write2txt(micarpeta + "StructuredOutput\\" + "CResidue" +".txt", GenWkt);
      }

    if (Vc) {
      Calendar datedemand = Calendar.getInstance();
      Calendar datetoday = Calendar.getInstance();
```

```
String aux8Wkt = "";
String aux81Wkt = "";
String aux82Wkt = "";
String myname = findmyindividuals(aux5[0],"hasDemand",bapron,manager);
String [] myname2 = myname.split(" ");
String Wkt = findmyindividuals(aux5[0],"hasID_Material",bapron,manager);
String [] auxWkt = Wkt.split(" ");
IRI mydemandID = IRI.create(bapron.getOntologyID().getOntologyIRI() + "#" +myname2[0]);
OWLIndividual mydemand = manager.getOWLDataFactory().getOWLNamedIndividual(mydemandID);
Map<OWLDataPropertyExpression,java.util.Set<OWLLiteral>> demandvalue = mydemand.getDataPropertyValues(bapron);
String demand = demandvalue.toString();
String[] demand2 = demand.split(",");
String[] demand21 = demand2[0].split("\"");
String[] demand22 = demand2[1].split("\"");
String[] demand221 = demand22[1].split("T");
String[] demand221A = demand221[0].split("-");
String[] demand221B = demand221[1].split(":");
int year = Integer.parseInt(demand221A[0]);
int month = Integer.parseInt(demand221A[1]);
int day = Integer.parseInt(demand221A[2]);
int hour = Integer.parseInt(demand221B[0]);
int minute = Integer.parseInt(demand221B[1]);
```

```
                        datedemand.set(year, month, day, hour, minute);
                        System.out.println(datetoday.getTime());

                        float hours = hoursBetween2Dates(datetoday,datedemand);

                        aux8Wkt = aux8Wkt + auxWkt[0] + "      " + demand21[1] + "&";
                        aux81Wkt = aux81Wkt + auxWkt[0] + "      " + demand22[1] + "&";
                        aux82Wkt = aux82Wkt + auxWkt[0] + "      " + hours + "&";

                        GenWkt5 = GenWkt5 + aux8Wkt;
                        GenWkt6 = GenWkt6 + aux81Wkt;
                        GenWkt11 = GenWkt11 + aux82Wkt;


                        write2txt(micarpeta + "StructuredOutput\\" + "demand" +".txt", GenWkt5);
                        write2txt(micarpeta + "StructuredOutput\\" + "duedate" +".txt", GenWkt6);
                        write2txt(micarpeta + "StructuredOutput\\" + "timehorizonhours" +".txt", GenWkt11);
                        }


//**********************************************************************
//   This part belongs to the SC optimization part
//**********************************************************************


                          This part of the code aims at finding demanded products.

                        if (Bh2) {
                        String myname = findmyvalues(aux5[0],"isDemanded",bapron,manager);
                        if (myname.equals ("true")){
                         DemandedProducts= DemandedProducts + aux5[0] + "&";
```

```
}
 write2txt(micarpetaSC + "StructuredOutput\\" + "EndProductD" + ".set", DemandedProducts);
}

if (Bh3) {
String myname = findmyvalues(aux5[0],"isDemanded",bapron,manager);
if (myname.equals ("true")){
 DemandedProducts= DemandedProducts + aux5[0] + "&";
}
 write2txt(micarpetaSC + "StructuredOutput\\" + "IntermediateD" + ".set", DemandedProducts);
}

if (Pll1 & findpll1)  {
    String auxpltP = "";
    String auxpltC = "";
    String auxcch = "";
    findpll1=false;
    String myname = findmyindividuals(aux5[0],"hasArea",bapron,manager);
    String[] auxkpn = myname.split(" ");
    String my2name = findmyindividuals(auxkpn[0],"hasProcessCell",bapron,manager);
    String[] auxkpn2 = my2name.split(" ");
    String my3name = findmyindividuals(auxkpn2[0],"hasUnit",bapron,manager);
    String[] auxkpn3 = my3name.split(" ");
    String my4name = "";
    String auxmincapacity="";
    String auxmaxcapacity="";

    for(int k=0; k<auxkpn3.length; k++){
```

```
                String mymincapacity=findmyvalues(auxkpn3[k],"min_value",bapron,manager);
                String mymaxcapacity=findmyvalues(auxkpn3[k],"max_value",bapron,manager);
                auxmincapacity=auxmincapacity + auxkpn3[k]+ "        " + mymincapacity + "&";
                auxmaxcapacity=auxmaxcapacity + auxkpn3[k]+ "        " + mymaxcapacity + "&";
                my4name = my4name + auxkpn3[k] + "&";


        }
        String my5name = findmyindividuals(aux5[0],"hasRecipe",bapron,manager);
        String[] auxkpn4 = my5name.split(" ");
        String my6name = findmyindividuals(auxkpn4[0],"derivesInMasterRecipe",bapron,manager);
        String[] auxkpn5 = my6name.split(" ");
        String my8name = "";
        for(int j=0; j<auxkpn5.length; j++){
        String my7name = findmyindividuals(auxkpn5[j],"hasRecipeElement",bapron,manager);
        String[] auxkpn6 = my7name.split(" ");
        for(int l=0; l<auxkpn6.length; l++){
        my8name = my8name + auxkpn6[l] + "&";
        }
    }

```

```
  /*INPUTS*/
  String[] recipeelements= my8name.split ("&");
  for(int j = 0; j < recipeelements.length; j++){
  String Gorr = findmyindividuals(recipeelements[j],"hasIDFromRecipeElementType",bapron,manager);
  String[] Gorr11 = Gorr.split(" ");
  String Gorr2 = findmyindividuals(recipeelements[j],"hasEquipmentRequirement",bapron,manager);
  String[] Gorr22 = Gorr2.split(" ");
  String Gorr3 = findmyindividuals(Gorr22[0],"referedFrom",bapron,manager);
  String Gorr4 = findmyindividuals(Gorr11[0],"receiveInformationFromIDRecipeElementType",bapron,manager);
  String[] Gorr44 = Gorr4.split(" ");
  String Gorr5 = findmyindividuals(Gorr44[0],"hasProcessInputParameter",bapron,manager);
  String[] Gorr55 = Gorr5.split(" ");

auxcch = auxcch +  Gorr3 + "." + recipeelements[j] + "     " + "1" + "&";

for(int k = 0; k < Gorr55.length; k++){
String Gorr551 = findmyindividuals(Gorr55[k],"hasParameterSource",bapron,manager);
String[] Gorr552 = Gorr551.split(" ");

IRI myqinID = IRI.create(bapron.getOntologyID().getOntologyIRI() + "#" +Gorr55[k]);
```

```
 OWLIndividual myqin = manager.
   getOWLDataFactory().
   getOWLNamedIndividual(myqinID);
 Map<OWLDataPropertyExpression,java
   .util.Set<OWLLiteral>> qinvalue
   = myqin.getDataPropertyValues(
   bapron);
String qin = qinvalue.toString();
String[] qin1 = qin.split("\"");
auxpltP = auxpltP + recipeelements[
  j] + "." + Gorr552[0] + "        "
  + qin1[1]  + "&";
}

/*OUTPUTS*/
String Gorr6 = findmyindividuals(
  Gorr44[0],"
  hasProcessOutputParameter",bapron
  ,manager);
String[] Gorr66 = Gorr6.split(" ");
for(int k = 0; k < Gorr66.length; k
  ++){
String Gorr551 = findmyindividuals(
  Gorr66[k],"hasParameterSource",
  bapron,manager);
String[] Gorr552 = Gorr551.split("
  ");
String Gorr5521 = findmyindividuals
  (Gorr552[0],"hasID_Material",
  bapron,manager);

IRI myqoutID = IRI.create(bapron.
  getOntologyID().getOntologyIRI()
  + "#" +Gorr66[k]);
OWLIndividual myqout = manager.
  getOWLDataFactory().
  getOWLNamedIndividual(myqoutID);
Map<OWLDataPropertyExpression,java.
  util.Set<OWLLiteral>> qoutvalue =
   myqout.getDataPropertyValues(
  bapron);
String qout = qoutvalue.toString();
String[] qout1 = qout.split("\"");
auxpltC = auxpltC + recipeelements[
  j] + "." + Gorr552[0] + "        "
  + qout1[1] +"&";
}

```

```
         }

          write2txt(micarpetaSC + "StructuredOutput\\" + "Technologies" + ".set", my4name);
          write2txt(micarpetaSC + "StructuredOutput\\" + "Tasks" + ".set", my8name);
          write2txt(micarpetaSC + "StructuredOutput\\" + "ProcessInputs" + ".txt", auxpltP);
          write2txt(micarpetaSC + "StructuredOutput\\" + "ProcessOutputs" + ".txt", auxpltC);
          write2txt(micarpetaSC + "StructuredOutput\\" + "tasks-tech" + ".txt", auxcch);
          write2txt(micarpetaSC + "StructuredOutput\\" + "mincapacitytech" + ".txt", auxmincapacity);
          write2txt(micarpetaSC + "StructuredOutput\\" + "maxcapacitytech" + ".txt", auxmaxcapacity);
          }


        if (Pll2) {
            String myname = findmyindividuals(aux5[0],"canCarryMaterialResource", bapron,manager);
            String [] mpc = myname.split(" ");
            String mymaxcapacity=findmyvalues(aux5[0] ,"max_value",bapron, manager);
            Maxcapacity=Maxcapacity + aux5[0]+ "        " + mymaxcapacity + "&";

            for(int j=0; j<mpc.length; j++){
                auxmpc=auxmpc + aux5[0] + "." + mpc[j] + "     1"+ "&";
            }
            write2txt(micarpetaSC + "StructuredOutput\\" + "trans-tech" + ".txt", auxmpc);
            write2txt(micarpetaSC + "StructuredOutput\\" + "capacitytransports" + ".txt", Maxcapacity);

        }

        if (Ptrr) {
             String myname = findmyindividuals(aux5[0],"hasFacility",bapron,manager);
             String [] mpc = myname.split(" ");

             for(int j=0; j<mpc.length; j++){
                 auxmpc=auxmpc + aux5[0] + "." + mpc[j] + "    1"+ "&";
             }
             write2txt(micarpetaSC + "StructuredOutput\\" + "facilitylocationrelationship" + ".txt", auxmpc);

        }

        if (Pch1){
             String myname = findmyindividuals(aux5[0],"relatesToProduct",bapron,manager);
             String [] mpc = myname.split(" ");
             auxmpc=auxmpc + aux5[0] + "." + mpc[0] + "    1"+ "&";
             write2txt(micarpetaSC + "StructuredOutput\\" + "transporte" + ".txt", auxmpc);
        }

        if (Pch2){
        String myname = findmyindividuals(aux5[0],"relatesToFacility",bapron,manager);
        String [] mpc = myname.split(" ");
        String myname2 = findmyvalues(aux5[0],"value",bapron,manager);
```

```
auxmpc=auxmpc +  mpc[0] + "    "+ myname2 + "&";
write2txt(micarpetaSC + "StructuredOutput\\" + "FacilityInvestmentCost" + ".txt", auxmpc);
}


if (Spt){
String myname = findmyindividuals(aux5[0],"demands",bapron,manager);
String [] mpc = myname.split(" ");
for(int j=0; j<mpc.length; j++){
String myname2 = findmyvalues(mpc[j],"value",bapron,manager);
String myname4 = findmyvalues(mpc[j],"salesPrice",bapron,manager);
String myname3 = findmyindividuals(mpc[j],"relatesToProduct",bapron,manager);
auxmpc=auxmpc + myname3+ "."+ aux5[0] + "    "+ myname2 + "&";
auxmpc2=auxmpc2 + myname3+ "."+ aux5[0] + "    "+ myname4 + "&";
}
write2txt(micarpetaSC + "StructuredOutput\\" + "SCDemand" + ".txt", auxmpc);
write2txt(micarpetaSC + "StructuredOutput\\" + "MarketPrice" + ".txt", auxmpc2);
}

if (Spt2){
String myname3 = findmyindividuals(aux5[0],"provideCost",bapron,manager);
String myname = findmyvalues(aux5[0],"value",bapron,manager);
auxmpc=auxmpc + myname3+ "    " + myname + "&";
write2txt(micarpetaSC + "StructuredOutput\\" + "CostRawMaterial" + ".txt", auxmpc);
}


```

```
if (FrE){
String myname3 = findmyindividuals(aux5[0],"relatesToFacility",bapron,manager);
String minvalue = findmyvalues(aux5[0],"min_value",bapron,manager);
String maxvalue = findmyvalues(aux5[0],"max_value",bapron,manager);
String value = findmyvalues(aux5[0],"value",bapron,manager);
String materials = findmyindividuals(aux5[0],"hasMaterialResource",bapron,manager);
String[] materials2 = materials.split(" ");
if (!maxvalue.equals("0.0")){
auxmpc2=auxmpc2 + myname3+ "     " + minvalue + "&";
auxmpc3=auxmpc3 + myname3+ "     " + maxvalue + "&";
}

if (!value.equals("0.0")){
for(int j=0; j<materials2.length; j++){
auxmpc=auxmpc +materials2[j] + "." + myname3+ "     " + value + "&";
}
}

write2txt(micarpetaSC + "StructuredOutput\\" + "minfacilitycapacity" + ".txt", auxmpc2);
write2txt(micarpetaSC + "StructuredOutput\\" + "maxfacilitycapacity" + ".txt", auxmpc3);
write2txt(micarpetaSC + "StructuredOutput\\" + "suppliercapacity" + ".txt", auxmpc);
}

if (FrE2){
 String auxpltP = "";
      String myname = findmyindividuals(aux5[0],"
                                    connectFacilities",bapron,
                                    manager);
                                  String[] auxkpn = myname.split
                                    (" ");
                                  String my2name =
                                    findmyindividuals(aux5[0] ,"
                                    hasTransportationCost",bapron
                                    ,manager);
                                  String[] auxkpn2 = my2name.
                                    split(" ");
                                  String my3name = findmyvalues(
                                    auxkpn2[0] ,"value",bapron,
                                    manager);
                                  String[] auxkpn3 = my3name.
                                    split(" ");
                                  auxkpon=auxkpon + auxkpn[0]+
                                    "." + auxkpn[1] + "         "
                                    + my3name + "&";
                                  auxkpon=auxkpon + auxkpn[1]+
                                    "." + auxkpn[0] + "         "
                                    + my3name + "&";

                                  write2txt(micarpetaSC + "
                                    StructuredOutput\\" + "
                                    TransportCosts" + ".txt",
                                    auxkpon);
                          }


                            if (Ptrr2) {
                                String myname = findmyindividuals
                                  (aux5[0],"hasFacility",bapron,
                                  manager);
                                String [] mpc = myname.split(" ")
                                  ;

                                for(int j=0; j<mpc.length; j++){
                                    auxmpc=auxmpc + aux5[0] + "."
                                        + mpc[j] + "     1"+ "&";
                                }
                                write2txt(micarpetaSC + "
                                  StructuredOutput\\" + "
                                  Marketlocationrelationship" +
                                  ".txt", auxmpc);

                          }

```

```
                  writetxt(micarpeta + "Output\\" + aux3[1]+".set", aux6);

            }
   }

        }
        catch (UnloadableImportException e) {
             If our ontology contains imports and one or more of the imports could not be loaded then an
             UnloadableImportException will be thrown (depending on the missing imports handling policy)
            System.out.println("Could not load import: " + e.getImportsDeclaration());
             The reason for this is specified and an OWLOntologyCreationException
            OWLOntologyCreationException cause = e.getOntologyCreationException();
            System.out.println("Reason: " + cause.getMessage());
        }
        catch (OWLOntologyCreationException e) {
            System.out.println("Could not load ontology: " + e.getMessage());
        }
    }

}
```

Appendix E

User's manual of Enterprise Ontology

This appendix presents the user's manual for instantiating any problem in the enterprise and process domain using the enterprise ontology.

E.1 Usage requirements

Next, the description of the elements necessary to start using the framework are listed:

1. A Java language editor, such as Eclipse or NetBeans, must be installed. This editor enables the management of the user's specific programming needs that can be managed from the Java enterprise ontology project. Either the user must have the programming language skills or any informatics programmer must be in charge of implementing or modifying the code for the connection between the ontological model and the decision-maker support tool.

2. The OWL Java API must be loaded within the Java editor, that is, the corresponding libraries must be added.

3. Prótegé software must be installed and ready to use. Otherwise, any full-OWL editor or manager should be installed. In addition, basic software skills are desired to be able to manage this platform. Alternatively, the Prótegé user's manual can be downloaded for more orientation.

4. The OWL file containing the enterprise ontology template (without any instance) located in the users workspace must be opened. Otherwise, if an already existing OWL file is used from past models, then such file must be opened for its modification.

E.2 Problem instantiation

The very first step for the problem instantiation consists of the identification and classification of the different elements that are involved in the reality of the process that the user aims to model. For this step, the use of the enterprise ontology classes definitions could help the user for an easier elements classification. In enterprise ontology, the classification is divided in nine principal parts, which are summarized next:

Processing activities Process operation and process actions are described in this class.

Process The description of the process as batch, discrete and continuous should be identified here.

Information The parameters and variables values related to supply chain management, process information, general information, recipe type, costs, identifiers, formula, process requirement, production parameters, procedural links, etc. are comprised as model information.

Physical model Process cell, unit, enterprise, equipment entity, control modules, areas, equipment modules and the different kind of facilities related to the physical description of the model are found here.

Process Output The lots and batches resulting from the production process are contained in this class.

Control Functions The information related to the coordinator control, equipment control, procedural control, procedural elements, etc. are defined within this class.

Production Process The process management, restrictions, control activities, inventories, demand, production process and other information directly related to production process must be addressed here.

Resources Transport, economic, human, unit, energetic and material resources should be classified within the different subclasses contained in resources.

Procedure This class contains the recipe procedure, equipment procedure, equipment unit procedure, etc.

E.3 Instances

Once that the enterprise ontology template is already loaded, the instantiation of the problem must be fulfilled according to the problem features. This action can be guided by the class/properties table (4.3) found in chapter 4, which provides the range and the domain of the relations that are established among classes.

The user can made the instantiation starting from top-down (starting from the main or general class to the required derived ones) or bottom-up (fulfilling first the classes that an expert user already knows that are required by a main or general class) perspective. The procedure for instantiating a given class is described next exemplifying the process for the master recipe class instantiation.

1. By making click on the Protégé (Protégé 3.4) individuals tab (Figure E.1, clue 1), the layout will show class browser (showing the class taxonomy), the instances browser (showing the asserted instances), and the individual editor.

2. Inside the class browser, recipe type class must be found clicking on master recipe class (Figure E.1, clue 2).

3. In order to make the first instance, the create instance button must be clicked, appearing a new instance that will be named MasterRecipe_1 as a default, renaming it as desired (Figure E.1, clue 3).

4. The enterprise ontology axioms will show the required fields that must be instantiated and are necessary for modeling the problem. The necessary fields are highlighted in red squares, as shown in Figure E.1.

5. Thus, the required fields of the instance must be fulfilled, within the individual editor. There are two possible ways to make it for an object property field:

   - Making click on the create resource button (if a top-down strategy was adopted) a new instance of the class contained in the range of the property must be created (Figure E.1, clue 4).
   - Otherwise, clicking the select existing resource button (if bottom-up strategy was adopted) adding an existing instance of the class contained in the range of the field property (Figure E.1, clue 5).

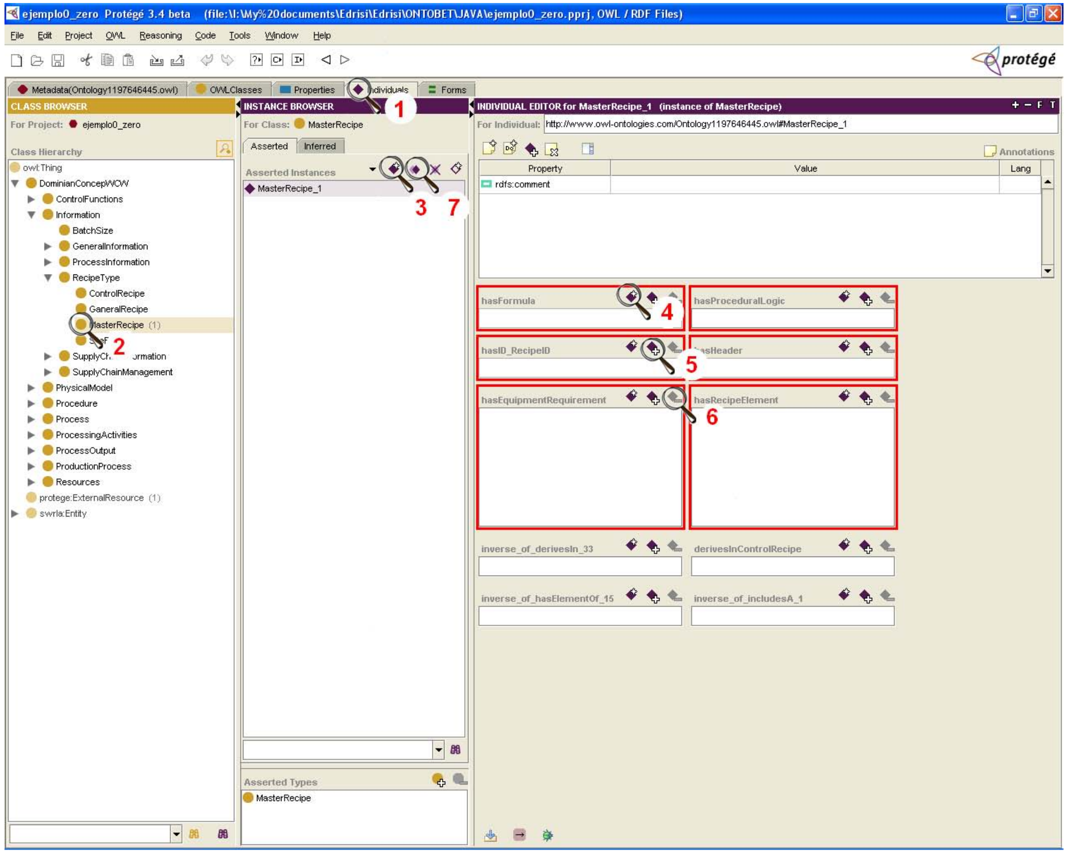
6. If the field requires more than one instance, just repeat the step number four.

7. If instance within a field is to be eliminated, select the instance clicking over it and then clock on the remove current value button (Figure E.1, clue 6).

8. If a another master recipe is to be instantiated, two alternatives can be followed.

   - Repeating step number three.
   - If there are more that one additional instance and is very similar to one already existing, the copy instance button should be used. (Figure E.1, clue 7), adding the number of copies as desired and just changing the right information for its differentiation.

9. Additionally, if a data object field is found, a data must be directly typed according to the correct range type (float, boolean, date, string, etc).



ejemplo0_zero Protégé 3.4 beta (file:\I:\My%20documents\Edrisi\Edrisi\ONTOBET\JAVA\ejemplo0_zero.pprj, OWL / RDF Files)

For Project: ejemplo0_zero

For Class: MasterRecipe

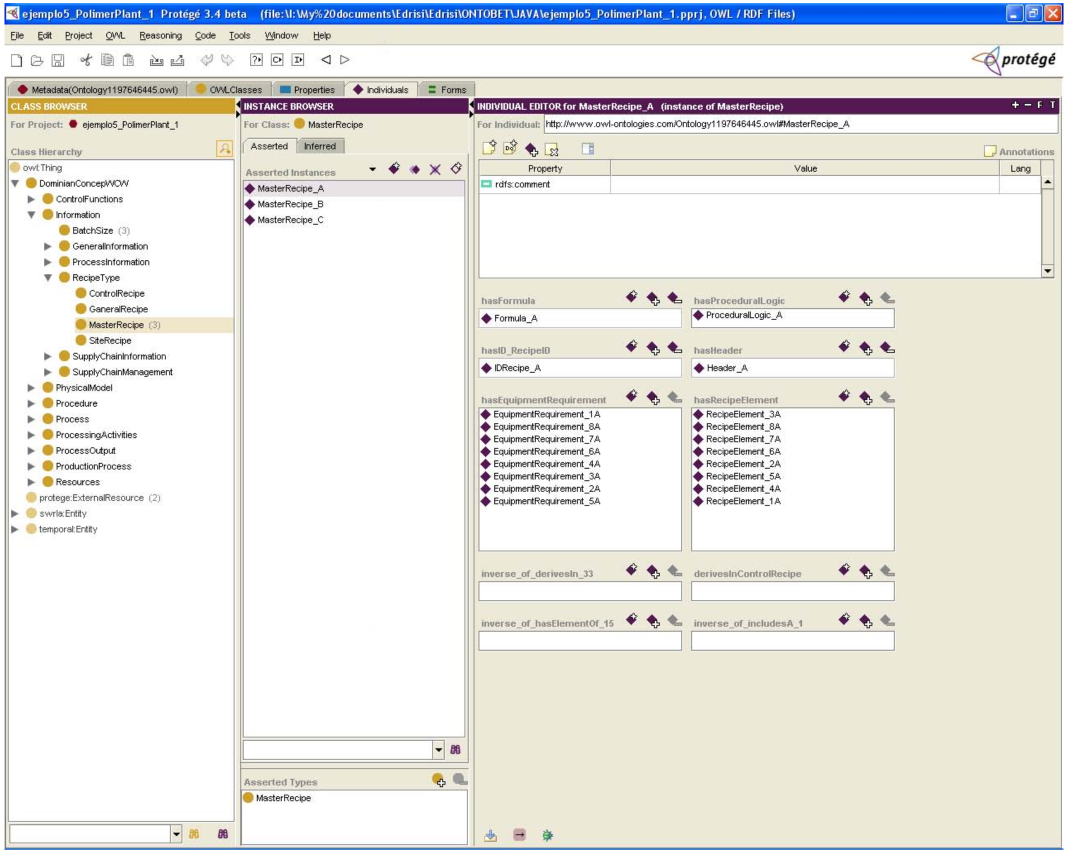
For Individual: http://www.owl-ontologies.com/Ontology1197646445.owl#MasterRecipe_1

Figure E.1: Screenshot of the enterprise ontology project template.

Next, for illustrating purposes the tree representations of the relationship derived from the master recipe and the site classes are shown.

For the master recipe case E.2:

- Formula
  - Parameter
    - value
    - Parameter Source (Material Resources)
- Header
  - Batch size
    - minimum value
    - maximum value
    - value
    - unit of measure
- Recipe ID
- Equipment requirement
  - Unit
  - ID Equipment requirement
- Procedural logic
  - ID logic
  - Link
    - Link type
    - ID Process stage (hasLinkFrom)
    - ID Process stage (hasLinkTo)
    - IDLink (provideID)
- Recipe element
  - Header
  - ID Recipe element
  - Equipment requirement
  - Process stage (has ID from Recipe Element)



ejemplo5_PolimerPlant_1 Protégé 3.4 beta (file:\I:\My%20documents\Edrisi\Edrisi\ONTOBET\JAVA\ejemplo5_PolimerPlant_1.pprj, OWL / RDF Files)

For Project: ejemplo5_PolimerPlant_1

Class Hierarchy

- owl:Thing
- DominianConcepWCW
  - ControlFunctions
  - Information
    - BatchSize (3)
    - GeneralInformation
    - ProcessInformation
    - RecipeType
      - ControlRecipe
      - GeneralRecipe
      - MasterRecipe (3)
      - SiteRecipe
    - SupplyChainInformation
    - SupplyChainManagement
  - PhysicalModel
  - Procedure
  - Process
  - ProcessingActivities
  - ProcessOutput
  - ProductionProcess
  - Resources
- protege:ExternalResource (2)
- swrla:Entity
- temporal:Entity

For Class: MasterRecipe

Asserted Instances

- MasterRecipe_A
- MasterRecipe_B
- MasterRecipe_C

Asserted Types

- MasterRecipe

INDIVIDUAL EDITOR for MasterRecipe_A (instance of MasterRecipe)

For Individual: http://www.owl-ontologies.com/Ontology1197646445.owl#MasterRecipe_A

Property	Value	Lang
rdfs:comment		

hasFormula: Formula_A

hasProceduralLogic: ProceduralLogic_A

hasID_RecipeID: IDRecipe_A

hasHeader: Header_A

hasEquipmentRequirement:
- EquipmentRequirement_1A
- EquipmentRequirement_8A
- EquipmentRequirement_7A
- EquipmentRequirement_6A
- EquipmentRequirement_4A
- EquipmentRequirement_3A
- EquipmentRequirement_2A
- EquipmentRequirement_5A

hasRecipeElement:
- RecipeElement_3A
- RecipeElement_8A
- RecipeElement_7A
- RecipeElement_6A
- RecipeElement_2A
- RecipeElement_5A
- RecipeElement_4A
- RecipeElement_1A

inverse_of_derivesIn_33

derivesInControlRecipe

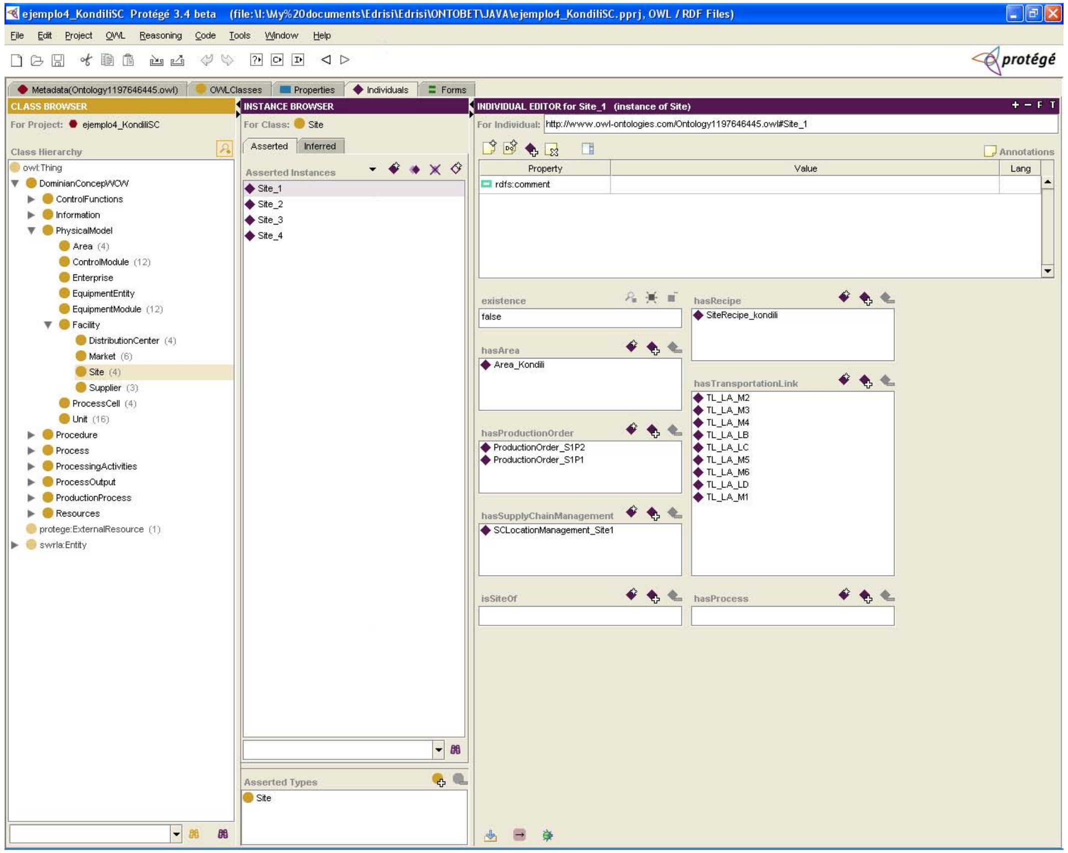
inverse_of_hasElementOf_15

inverse_of_includesA_1

Figure E.2: Screenshot of the master recipe instantiation.

For the site case E.3:

- Area
  - Process Cell
- Production Order
  - Duedate
  - Value
  - Unit of measure
  - Material Resource
- Site Recipe
- Supply Chain Management
  - Supply Chain Location Management
    - Is applied at (Facility)
    - Has location cost
      - Facility investment cost
    - Has location parameter
      - Has facility (Facility)
  - Supply Chain Inventory Management
  - Supply Chain Production and Distribution Management
  - Supply Chain Transportation Management
- Transportation links
  - Connect facilities (Facility)
  - Has transport (Transport resources)
  - Has transportation cost
  - Transport material (ID material resources)



ejemplo4_KondiliSC Protégé 3.4 beta (file:\I:\My%20documents\Edrisi\Edrisi\ONTOBET\JAVA\ejemplo4_KondiliSC.pprj, OWL / RDF Files)

CLASS BROWSER

For Project: ejemplo4_KondiliSC

Class Hierarchy

- owl:Thing
- DominianConcepWCW
  - ControlFunctions
  - Information
  - PhysicalModel
    - Area (4)
    - ControlModule (12)
    - Enterprise
    - EquipmentEntity
    - EquipmentModule (12)
    - Facility
      - DistributionCenter (4)
      - Market (6)
      - Site (4)
      - Supplier (3)
    - ProcessCell (4)
    - Unit (16)
  - Procedure
  - Process
  - ProcessingActivities
  - ProcessOutput
  - ProductionProcess
  - Resources
- protege:ExternalResource (1)
- swrla:Entity

INSTANCE BROWSER

For Class: Site

Asserted Instances

- Site_1
- Site_2
- Site_3
- Site_4

Asserted Types

- Site

INDIVIDUAL EDITOR for Site_1 (instance of Site)

For Individual: http://www.owl-ontologies.com/Ontology1197646445.owl#Site_1

Property	Value	Lang
rdfs:comment		

existence: false

hasArea:
- Area_Kondili

hasProductionOrder:
- ProductionOrder_S1P2
- ProductionOrder_S1P1

hasSupplyChainManagement:
- SCLocationManagement_Site1

isSiteOf:

hasRecipe:
- SiteRecipe_kondili

hasTransportationLink:
- TL_LA_M2
- TL_LA_M3
- TL_LA_M4
- TL_LA_LB
- TL_LA_LC
- TL_LA_M5
- TL_LA_M6
- TL_LA_LD
- TL_LA_M1

hasProcess:

Figure E.3: Screenshot of the site instantiation.

E.4 Instances inference

After the complete instantiation of the problem, it is necessary to compute the inferred types. This step provides feedback to the user about the logical implications of the model design.

Within the menu "Reasoning", the user should select the desired reasoner and infer the instances by clicking on "Compute inferred types...". A window with the information of the reasoner's process will appear.

E.5 Framework exploitation

In order to exploit the ontological model, it is necessary to program the Java interface. The result of this process will provide the correct data as appropriate to the different objectives for which the ontology is adopted. Next a brief example of how it works will be provided.

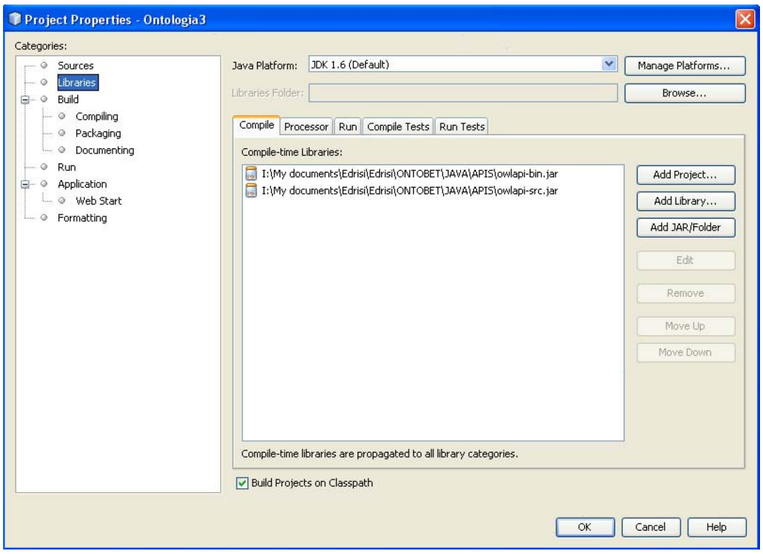
1. Load libraries. Within the project properties it is necessary to load the OWL Java APIS *owlapi-bin* and *owlapi-src*, in order to work with the functions in the main Java project (Figure E.4).

2. Create the main Java File. The next step consist of creating a main Java file which contains the following parts:

    - Import the OWL-API classes which are to be used within the code (Figure E.5).
    - Declare path of the workspace.
    - Declare and load the OWL file already instantiated (Figure E.6).

3. Create the necessary variables according to the decision support tool requirements (Figure 5.3).

4. Structure the form of the desired output file and give them the right extension.



Project Properties - Ontologia3

Java Platform: JDK 1.6 (Default)

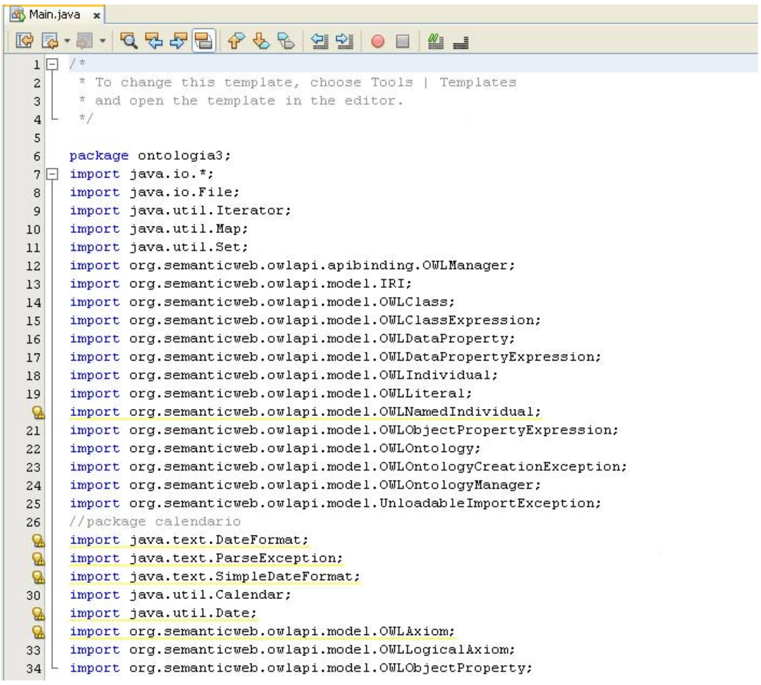
Compile-time Libraries:

I:\My documents\Edrisi\Edrisi\ONTOBET\JAVA\APIS\owlapi-bin.jar
I:\My documents\Edrisi\Edrisi\ONTOBET\JAVA\APIS\owlapi-src.jar

Compile-time libraries are propagated to all library categories.

☑ Build Projects on Classpath

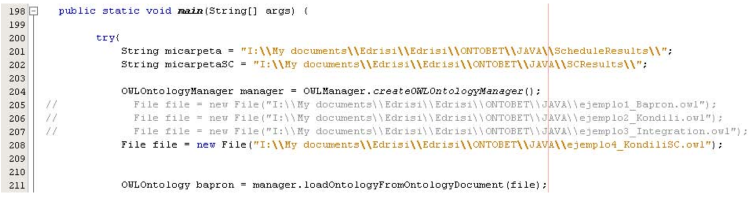
Figure E.4: Step 1. Libraries loading.



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package ontologia3;
import java.io.*;
import java.io.File;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLClassExpression;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLDataPropertyExpression;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLLiteral;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectPropertyExpression;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.UnloadableImportException;
//package calendario
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import org.semanticweb.owlapi.model.OWLAxiom;
import org.semanticweb.owlapi.model.OWLLogicalAxiom;
import org.semanticweb.owlapi.model.OWLObjectProperty;
```

Figure E.5: Step 2. OWL classes importation.



```
    public static void main(String[] args) {

        try{
            String micarpeta = "I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\ScheduleResults\\";
            String micarpetaSC = "I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\SCResults\\";

            OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
//            File file = new File("I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\ejemplo1_Bapron.owl");
//            File file = new File("I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\ejemplo2_Kondili.owl");
//            File file = new File("I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\ejemplo3_Integration.owl");
            File file = new File("I:\\My documents\\Edrisi\\Edrisi\\ONTOBET\\JAVA\\ejemplo4_KondiliSC.owl");


            OWLOntology bapron = manager.loadOntologyFromOntologyDocument(file);
```

Figure E.6: Step 2. Work space declaration and OWL file loading.

Bibliography

Alavi, M. and P. G. W. Keen (1989). Business teams in an information age. *Inf. Soc. 6*(4), 179–95.

Apostolou, D., G. Mentzas, and A. Abecker (2008, june). Ontology-enabled knowledge management at multiple organizational levels. In *Engineering Management Conference, 2008. IEMC Europe 2008. IEEE International*, pp. 1 –6.

Avgeriou, P. and U. Zdun (2005). Architectural patterns revisited: A pattern language. In *In 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee*, pp. 1–39.

Bechhofer, S., I. Horrocks, C. Goble, and R. Stevens (2001). OilEd: A reason-able ontology editor for the semantic Web. *LNCS 2174*, 396ff.

Bechhofer, S., F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. (2004, February). Owl web ontology language.

Bernaras, A., I. Laresgoiti, and J. Corera (1996). Building and reusing ontologies for electrical network applications. In W. Wahlster (Ed.), *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 96): Budapest, Hungary: August 11-16*, pp. 298–302. Wiley.

Borgo, S., N. Guarino, and C. Masolo (1996). Stratified ontologies: The case of physical objects.

Bosak, J. (1997). Xml, java, and the future of the web. *World Wide Web Journal*, 219–227.

Brandl, D. and D. Emerson (2003, September). Batch markup language batchml.

Brickley, D. and R. Guha (2002, November). Rdf vocabulary description language 1.0: Rdf schema.

CEFIC, E. C. I. C. (2009). High level group on the competitiveness of the european chemicals industry final report.

Chopra, S. and P. Meindl (2004). *Supply chain management : strategy, planning, and operation.* Prentice Hall.

Council, T. E. C. I. (2010). Cefic review 2009-2010. sustainability and innovation driving chemistry solutions for the future. Technical report, CEFIC.

Courtney, J. F. (2001). Decision making and knowledge management in inquiring organizations: toward a new decision-making paradigm for dss. *Decis. Support Syst. 31*(1), 17–38.

Davis, R., H. Shrobe, and P. Szolovits (1993). What is a knowledge representation?

Deming, W. E. (1993). *The New Economics.* Massachusetts Institute of Technology Press.

Domingue, J. (2010, June). Webonto.

Eppler, M. J. (2006). *Introducing the Notion of Information Quality* (second ed.). Springer Berlin Heidelberg.

Farquhar, A., R. Fikes, and J. Rice (1997). The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies 46*, 707–727.

Fensel, D. (2003, December). *Ontologies: : A Silver Bullet for Knowledge Management and Electronic Commerce.* Springer.

Fernandez-Breis, J. T. and R. Martinez-Bejar (2000). A cooperative tool for facilitating knowledge management. *Expert Systems with Applications 18*(4), 315 – 330.

Fernandez-Lopez, M., A. Gomez-Perez, and N. Juristo (1997, March). Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium*, Stanford, USA, pp. 33–40.

Informatics Research for Biomedical(BMIR), S. C. (2007). Protègè 3.4 and protègè 4.1.

International Society for Measurement and Control(1995, October). Batch control part 1 models and terminology.

International Society for Measurement and Control (1999). Enterprise control system integration part 1 models and terminology.

International Society for Measurement and Control (2003, March). Batch control part 3: General and site recipe models and representation.

International Society for Measurement and Control(2006, Agosto). Control batch parte 4 registros de producción de lote.

International Society for Measurement and Control (2007a, January). Batch control part 5 automated equipment control models & terminology.

International Society for Measurement and Control (2007b, December). Isa-88/95 technical report: Using isa-88 and isa-95 together. Technical report, ISA The Instrumentation, Systems, and Automation Society.

Fox, M. S., M. Barbuceanu, M. Gruninger, and J. Lin (1997). An organization ontology for enterprise modelling. In *Modeling, In: International Conference on Enterprise Integration Modelling Technology 97.* Springer.

Fox, M. S. and M. Gruninger (1998). Enterprise modeling. *AI Magazine 19*(3), 109–121.

Gebus, S. and K. Leiviskä (2009). Knowledge acquisition for decision support systems on an electronic assembly line. *Expert Syst. Appl. 36*(1), 93–101.

Gómez-Pérez, A. (2007). Ontological engineering: A state of the art.

Gómez-Pérez, A. and M. Fernández (1996). Towards a method to conceptualize domain ontologies. In *Proceedings of the workshop on Ontological Engineering.* ECAI.

Gómez-Pérez, A. and M. C. Suárez-Figueroa (2003). Results of taxonomic evaluation of rdf(s) and daml+oil ontologies using rdf(s) and daml+oil validation tools and ontology platforms import services. In *II ISWC Workshop on Evaluation of*

Ontology-based Tools, Volume 87.

Gosling, J., B. Joy, G. Steele, and G. Bracha (2005). *Java(TM) Language Specification, The (3rd Edition) (Java (Addison-Wesley))*. Addison-Wesley Professional.

Gruber, T. R. (1993, June). A translation approach to portable ontology specifications. *Knowl. Acquis. 5*(2), 199–220.

Grubic, T. and I.-S. Fan (2010, June). Supply chain ontology: Review, analysis and synthesis. *Computers in Industry*.

Gruninger, M. and M. Fox (1995). Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*.

Holmes, J. (2004). *Struts: The Complete Reference (Osborne Complete Reference Series)*. McGraw-Hill Osborne Media.

Horridge, M., S. Bechhofer, and O. Noppens (2007). Igniting the owl 1.1 touch paper: The owl api. In *OWLED 2007, 3rd OWL Experienced and Directions Workshop*.

Horridge, M., S. Jupp, G. Moulton, A. Rector, R. Stevens, and C. Wroe (2007). A practical guide to building owl ontologies using protege 4 and co-ode tools. Technical report, The University Of Manchester.

Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta (2000). Ontology inference layer oil.

Johanson, L. (2011, June). Freecbr.

Karp, P., K. Myers, and T. Gruber (1995). The generic frame protocol. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence (IJCAI95)*, Montreal, Canada., pp. 768–774.

Klein, M. R. and L. B. Methlie (1995). *Knowledge-Based Decision Support Systems: With Applications in Business* (2nd ed.). New York, NY, USA: John Wiley & Sons, Inc.

Klyne, G. and J. J. Carroll (2002, November). Resource description framework (rdf): Concepts and abstract syntax.

Knight, K. and S. Luk (1994). Building a large knowledge base for machine translation. In *Proceedings of the American Association of Artificial Intelligence Conference*, Seattle, USA, pp. 773–778.

Kondili, E., C. Pantelides, and R. Sargent (1993). A general algorithm for short-term scheduling of batch operations–i. milp formulation. *Computers & Chemical Engineering 17*(2), 211 – 227.

Korovessi, E. and A. A. Linninger (2006). *Batch processes*. Taylor & Francis Group.

Lainez, J. M., G. Kopanos, A. Espuna, and L. Puigjaner (2009, JUL). Flexible design-planning of supply chain networks. *AICHE JOURNAL 55*(7), 1736–1753.

Lenat, D. B. and R. V. Guha (1989). *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project* (1st ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Liao, S.-H. (2003). Knowledge management technologies and applications–literature review from 1995 to 2002. *Expert Systems with Applications 25*(2), 155 – 164.

Linwood, J., D. Minter, J. Linwood, and D. Minter (2010). Integrating and configuring hibernate. In *Beginning Hibernate*, pp. 9–25. Apress.

López, M. F., A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE*

Intelligent Systems 14(1), 37–46.

MacGregor, R. (1991). *Principles of semantic networks*, Chapter The evolving technology of classification-based knowledge representation systems., pp. 385–400. Morgan Kaufmann.

Maravelias, C. T. and I. E. Grossmann (2003). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research 42*, 3056–3074.

Mathworks (2009). The mathworks corporation. matlab 7.2. Technical report, http://www.mathworks.com.

McQueen, S. and H. Thompson (2000, April). Xml schema.

Measurement, I. S. F. and Control (2001, February). Data structures and guidelines for languages.

Mendez, C. A., J. Cerda, I. E. Grossmann, I. Harjunkoski, and M. Fahl (2006, May). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering 30*(6-7), 913–946.

Missikoff, M. and F. Taglino (2002, January). Business and enterprise ontology management with symontox. In S. B. . Heidelberg (Ed.), *The Semantic Web - ISWC 2002*, Volume 2342/2002, pp. 442–447.

Mitroff, I. I. and H. A. Linstone (1993). The unbounded mind: Breaking the chains of traditional business thinking. *Business Horizons 36*(5), 88–89.

Mizoguchi, R., J. Vanwelkenhuysen, and M. Ikeda (1995). *Task Ontology for Reuse of Problem Solving Knowledge*. IOS press.

Morris, W. T. (1967). On the arts of modelling. *Management Science 13*, 707–717.

Obrst, L. (2003, November). Ontologies for semantically interoperable systems. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pp. 366–369. ACM.

Pagels, M. (2006, January). The darpa agent markup language (daml).

Pekny, J. F. and G. V. Reklaitis (1998). *Towards the Convergence of Theory and Practice: A Technology Guide for Scheduling/Planning Methodology*. Foundations of Computer Aided Process Operations.

Power, D. J. and R. Sharda (2007). Model-driven decision support systems: Concepts and research directions. *Decis. Support Syst. 43*(3), 1044–1061.

Raleigh, T. and D. Harmelink (2004). *The Supply chain handbook*. Tompkins Press.

Rippin, D. (1983). Batch process systems - engineering: A retrospective and prospective review. *Computers & Chemical Engineering 17*, S1 – S13.

Schreiber, G., H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, MA: MIT Press.

Shim, J. P., Warkentin, Merrill, Courtney, F. James, Power, J. Daniel, Sharda, Ramesh, Carlsson, and Christer (2002). Past, present, and future of decision support technology. *Decis. Support Syst. 33*(2), 111–126.

Shirasuna, M. (2007, June). Optimización de la producción en una empresa de cervezas usando el estándar isa s88 (caso polar). In *Rockwell Automatization*.

Simon, F. and T. Murray (2007). Decision support systems. *Commun. ACM 50*(3), 39–40.

Standford-University (2010). Webonto.

Subrahmanyam, S., M. Bassett, J. Pekny, and G. Reklaitis (1995). Issues in solving large scale planning, design and scheduling problems in batch chemical plants. *Computers and Chemical Engineering 19*, 577 – 582.

Sure, Y., M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke (2002). Ontoedit: Collaborative ontology development for the semantic web. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, London, UK, UK, pp. 221–235. Springer-Verlag.

Sure, Y. and R. Studer (2002, September). On-to-knowledge methodology. Technical report, University of Karlsruhe.

Table, M. A. (2004). Mysql reference manual.

Uschold, M. and M. Gruninger (1996). Ontologies: principles, methods, and applications. *Knowledge Engineering Review 11*(2), 93–155.

van Heijst, G., A. T. Schreiber, and B. J. Wielinga (1997). Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud. 46*(2-3), 183–292.

Varma, V. A., G. V. Reklaitis, G. E. Blau, and J. F. Pekny (2007, May). Enterprise-wide modeling & optimization - an overview of emerging research challenges and opportunities. *Computers & Chemical Engineering 31*(5-6), 692–711.

Vega, J. C. A., A. Gómez-Pérez, A. L. Tello, H. S. A. N. P. Pinto, H. Sofia, and A. N. P. Pinto (1996). (onto)&sup2;agent: An ontology-based www broker to select ontologies. In *Workshop on application Ontologies and PSMs*, pp. 16–24.

Venkatasubramanian, V., C. Zhao, G. Joglekar, A. Jain, L. Hailemariam, P. Suresh, P. Akkisetty, K. Morris, and G. Reklaitis (2006, July). Ontological informatics infrastructure for pharmaceutical product development and manufacturing. *Computers and Chemical Engineering 30*, 1482–1496.

Williams, T. J. (1989). *A reference model for computer integrated manufacturing (CIM)*. ISA Research Triangle Park.

XML-Core-Working-Group (2009, April). Extensible markup language (xml).