August 2018

A Knowledge Enriched Computational Model to Support Lifecycle Activities of Computational Models in Smart Manufacturing

Heng Zhang
Syracuse University

ABSTRACT

Due to the needs in supporting lifecycle activities of computational models in Smart Manufacturing (SM), a Knowledge Enriched Computational Model (KECM) is proposed in this dissertation to capture and integrate domain knowledge with standardized computational models. The KECM captures domain knowledge into information model(s), physics-based model(s), and rationales. To support model development in a distributed environment, the KECM can be used as the medium for formal information sharing between model developers. A case study has been developed to demonstrate the utilization of the KECM in supporting the construction of a Bayesian Network model. To support the deployment of computational models in SM systems, the KECM can be used for data integration between computational models and SM systems. A case study has been developed to show the deployment of a Constraint Programming optimization model into a Business To Manufacturing Markup Language (B2MML) -based system. In another situation where multiple computational models need to be deployed, the KECM can be used to support the combination of computational models. A case study has been developed to show the combination of an Agent-based model and a Decision Tree model using the KECM. To support model retrieval, a semantics-based method is suggested in this dissertation. As an example, a dispatching rule model retrieval problem has been addressed with a semantics-based approach. The semantics-based approach has been verified and it demonstrates good capability in using the KECM to retrieve computational models.

A KNOWLEDGE ENRICHED COMPUTATIONAL MODEL TO SUPPORT LIFECYCLE ACTIVITIES OF COMPUTATIONAL MODELS IN SMART MANUFACTURING

By

Heng Zhang

B.S., Hebei University of Technology, 2010
M.S., Syracuse University, 2012

Dissertation
Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Mechanical and Aerospace Engineering

Syracuse University

August 2018

ACKNOWLEDGMENTS

First, I would like to express my deepest appreciation to my advisor, Dr. Utpal Roy, for his continuous help in guiding me throughout my doctoral studies. I am also very grateful for the opportunities for international conferences, NIST research workshop, and industrial company collaboration that Dr. Roy provided to make me a better researcher.

I also would like to thank Dr. Riyad S. Aboutaha, Dr. Jianshun Zhang, Dr. Jeffrey S. Saltz, Dr. Michael N. Roppo, and Dr. Teng Zhang for serving as my committee members and providing valuable suggestions and comments for this dissertation.

A big “thank you” goes to Dr. Bicheng Zhu for all his help in fundamental research problem discussions, encouragement, and suggestions for life. Further thanks go to the lab colleagues and friends, including Omer Yaman, Yunpeng Li, Hang Yin, and Kai Sun, for their good suggestions in my research work.

Special thanks go to Dr. Xianrui Wang and Dr. Xu Yan, for their friendship and support across the world. I also want to thank my friends Dr. Jing Wang, Zhengyi Song, and Daiyang Gao for their companionship in graduate school.

Finally, I would like to acknowledge the most important persons in my life – my parents Zhijian Zhang and Huiying Jia for their unconditional love. Without their support, I could not overcome all the challenges I faced all these years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. Introduction

In this chapter, an overview of the research in this dissertation is presented. This chapter starts with research motivation. The research objective and proposed methodology are then introduced. Finally, the structure of the dissertation is outlined.

1.1 Research Motivation

Due to advances in information technologies and artificial intelligence, the Smart Manufacturing (SM) concept has emerged to lead a new paradigm of manufacturing. The SMLC (Smart Manufacturing Leadership Coalition) has characterized the SM enterprises as data-driven, knowledge-enabled, and model rich with visibility across the enterprise, such that all operating actions are executed proactively by applying the best information and performance metrics (Davis et al., 2015). To achieve this, computational models to be easily accessible and available to a wide range of users across enterprises (SMLC., 2011). According to the SMLC, this calls for the standardization of computational models to support plug-and-play capability and effective data exchange for industrial users from small manufacturing enterprises to large ones. It also requires human knowledge and decisions to be incorporated into decision models, which enables faster, more disciplined decision making.

Computational models, which are the core components of enterprise decision tools, play important roles in both business and engineering decision making at all levels of an enterprise's hierarchy from business planning and logistics through manufacturing operations control to batch and unit process control. Here, the computational models can be the mathematical,

optimization, knowledge-based, data analytics/machine learning and rule models, etc. that are used in enterprise decision making. To support their interoperability/accessibility, standardized computational models have been developed to formally represent computational models in text-based formats like XML (Extensible Markup Language) and JSON (JavaScript Object Notation). For example, Mathematical Markup Language (MathML) (W3C, 2004) has been developed to represent mathematical expressions using XML. The Predictive Model Markup Language (PMML) (DMG, 2016) is a set of XML-based data models to represent statistical and data mining models.

However, the current standardized computational models do not possess formally captured domain knowledge which can be used to support the lifecycle activities of computational models. The lifecycle activities that are focused in this dissertation are the development, deployment, and retrieval of computational models. Here, the knowledge can be the domain meanings of the entities in computational models, the physics or behavioral information about the application domain where a computational model applies, the rationales or rules to describe the rationality of a computational model or to guide the lifecycle activities of computational models. These types of knowledge are very important to support lifecycle activities of computational models. The domain meanings of a computational model's entities are needed in all its lifecycle activities so that the computational model can be understood. Physics or behavioral information about the application domain of a computational model can be used to support the development of the computational model. Rationales or rules can capture information about why a specific structure of a computational model was developed or how a model parameter was defined, which are

useful when the computational model needs to be modified or updated. The rules can also be used to guide the development, deployment, and retrieval of computational models. All these types of knowledge need to be formally captured and integrated with the standardized computational models so that software tools can be used to automate lifecycle activities of computational models.

So, to support lifecycle activities of computational models, a methodology has been proposed in this dissertation to formally capture knowledge and integrate the knowledge with standardized computational models.

1.2 Research Objective and Proposed Methodology

According to the research motivation introduced in the last section, the research objective of this dissertation is *to develop a knowledge enriched computational model which formally captures knowledge and integrates the knowledge with standardized computational models to support lifecycle activities of computational models*.

The proposed methodology in this dissertation is as follows:

A Knowledge Enriched Computational Model (KECM) has been proposed to capture knowledge into information model(s), physics-based model(s) and rationales. The information model(s) can be used to capture domain concepts and relationships. The physics-based model(s) can be used to encapsulate the physical or behavioral information of a certain SM system (e.g., a manufacturing process, a shop floor control system, a business planning system, etc.). The rationales or rules can be used to provide rationality about computational models and to guide

lifecycle activities. Semantic links can be used to integrate these types of knowledge with the standardized computational models.

An example of the KECM for developing a Bayesian Network (BN) model is provided as follows.

A BN model needs to be developed for estimating the energy consumption of injection molding processes. Due to the lack of formal information exchange between domain experts and data analysts, a KECM which can formally represent the BN model and domain knowledge can be used to support the formal information exchange. To formally represent the BN model, the PMML – Bayesian Network Model (DMG, 2016) can be used as the standardized computational model in the KECM. To capture domain meanings for the nodes in the BN, a Process-oriented Information Model for Sustainable Manufacturing from a literature (Zhang et al., 2015) can be used for the KECM. To generate the structure of the BN based on domain knowledge, mathematical equations which calculate the energy consumption for injection molding processes can be used. The OntoModel can be used to formally represent these equations as the physics-based models in the KECM. Rationales/rules can be developed to describe how to use the OntoModel-based equations to generate the BN structure. A detailed case study of this example has been provided in Chapter 4.

The proposed KECM has been further validated in three distinct applications to demonstrate the utilization of the KECM to support three different lifecycle activities for computational models. The three applications are as follows:

(1) *implementing a KECM to support the development of a Bayesian Network model in a distributed environment,*

(2) *implementing KECMs to support the deployment of computational models: the deployment of an optimization model in a manufacturing system and the combination of an Agent-based model and a Decision Tree model for a real-time scheduling scenario, and*

(3) *developing KECM rationales/rules that formally describe dispatching rule models for the retrieval of dispatching rules.*

With the proposed methodology, outcomes to be expected are:

(1) *For computational models that need to be developed in distributed environments, the proposed methodology enables explicit and formal knowledge exchange between model developers and further enhances the efficiency of distributed model development.*

(2) *The proposed methodologies enable industrial users to define their own ways to deploy and combine computational models in Smart Manufacturing systems.*

(3) *The proposed methodology allows the retrieval of computational models according to users' requirements for Smart Manufacturing applications.*

1.3 Dissertation Outline

This dissertation is presented in seven chapters. The description of each chapter is narrated below.

Chapter 2 reviews literature related to the research problem. It first reviews literature that formally represents computational models and their relevant knowledge to support model

interoperability. The next section reviews literature that formalizes domain knowledge to support lifecycle activities of computational models. This section is divided into three sub-sections. Each sub-section reviews literature related to a specific lifecycle activity – model development, model deployment, and model retrieval.

Chapter 3 proposes a Knowledge Enriched Computational Model (KECM) that explicitly and formally enriches domain knowledge into standardized computational models to support the lifecycle activities of computational models. The general description of the KECM is provided.

Chapter 4 discusses the utilization of the proposed KECM to support model development. A general method to use the KECM in model development has been proposed. To validate the proposed method, a case study has been presented to develop a Bayesian Network model with the assistance of the KECM. The KECM for the Bayesian Network model has been developed. The utilization of the KECM in supporting the development of the Bayesian Network model has been discussed.

Chapter 5 discusses the utilization of the proposed KECM to support model deployment. This chapter is divided into two parts. The first part covers the general method that using the KECM to support the data integration between a computational model and the data system that the computational model deploys. A case study has been provided to demonstrate the deployment of an optimization model in a B2MML-based data system. Due to the lack of a standardized model for optimization models, an Optimization Metamodel has been proposed. In the second part, a general approach to formally represent model combinations is introduced. This model combination method aims to extend the standardized computational model captured in the

KECM. As an example, a combination of an Agent-based model and a Decision Tree has been carried out using the proposed modeling method.

Chapter 6 describes a method to retrieve computational models using the KECM. In this chapter, the retrieval of dispatching rule models based on production objectives has been studied as an example. To enable this retrieval, a semantics-based approach has been proposed. This semantics-based approach first defines the formal semantic expression of dispatching rules and production objectives. Then, a tree-based semantic similarity measurement has been presented to calculate the similarities between the given production objectives and all the dispatching rules. Based on the similarity values, the selected dispatching rule model can be combined. The validation of the semantics-based dispatching rule selection approach has been provided at the end of the chapter.

Chapter 7 concludes this dissertation. It starts with a summary of the whole dissertation. Then, the research contributions are described. Finally, the limitations of the research work in the dissertation are discussed.

CHAPTER 2. Literature Review

In this chapter, the literature related to the research problem that this dissertation focuses on is reviewed. First, the literature that is related to formally representing computational models and relevant knowledge is reviewed. Then, the literature that is related to formalizing knowledge to support model development, deployment and retrieval, respectively are reviewed.

2.1 Formally Representing Computational Models and Relevant Knowledge to Support Model Interoperability

To enable the accessibility and plug-and-play capability of computational models, computational models must be formally represented to support their interoperability. Currently, there are several industry standards that have been developed to support certain types of computational models. For example, the MathML (W3C, 2004) is an XML-based language to represent mathematical expressions. The MathML has two parts: the ContentML is used to represent the meanings of mathematical expressions, and the PresentationML is used to capture the presentation of mathematical expressions (e.g., matrix, vector, and tables, etc.). By using MathML, mathematical expressions can be smoothly shared in web applications and text editing tools like Microsoft Office. The PMML (DMG, 2016) is also an XML-based language to formally represent statistical and data mining models. Models like Regression, Decision Tree, Neural Network, and Bayesian Network, etc. can be represented in XML-based documents to allow model exchange between different data analytics applications. The ECSS E-TM-40-07/Simulation Model Portability 2 (ECSS, 2011) is a standard developed for

representing simulation models. Mappings of the metamodels, component models and simulation services to the C++ platform have been developed.

There are also academic efforts to formally represent computational models and their relevant knowledge. Witherell et al. (2007) presented an ontology for optimization (ONTOP) in the engineering design domain to bridge the semantic gap in inconsistent optimization terminologies across different software tools. Though formal semantics were used to capture the overall structure and terminology of optimization models, entities like objective function and constraints are represented in strings. This informal representation of mathematical expressions created issues for effective data exchange. One issue raised is that there is a need to integrate optimization models with knowledge like the engineers' rationales in creating the models. Thus, to capture the engineering design knowledge related to optimization models, the authors proposed to include assumptions, model purposes, and related images, etc. in the ontology. However, the assumptions, model descriptions, and associated images were captured either in natural language-based strings or in formats that are difficult for software tools to process. Although the developed ontology was implemented using the Web Ontology Language (OWL), it is difficult for software tools to process and understand a whole optimization model. Moreover, the parsing and utilization of the embedded domain knowledge rely on manual work.

Muñoz et al. (2014) proposed an ontological approach to represent optimization models and manufacturing information. The representation of mathematical programming models was achieved by using their previously developed Ontological Math Representation (Muñoz et al., 2012) ontological model. The manufacturing information was captured by their Enterprise

Ontology Project (Muñoz et al., 2013) ontological model. However, there is no clear description about how to use their model to represent optimization models and how to integrate the two ontological models. Additionally, only information models (i.e. ontological model) were used as domain knowledge.

Denno and Kim (2016) proposed a semantic web technology-based idea to integrate predictive model equations (i.e. regression models) in unit manufacturing process models. The authors argued that it is valuable to share predictive model equations among industrial users because the equations reflect certain manufacturing knowledge that can be used elsewhere. They utilized semantic web technology to semantically connect objects in equations to manufacturing concepts. They claimed the proposed idea could benefit knowledge refinement and reuse, traceability, model verification in production activities. However, there was no formal representation of the equations shown to support the inoperability of the equations. Also, there were no clear semantic links between the equation variables and manufacturing concepts demonstrated in the paper. There was also no work to show how the proposed method supports downstream activities (e.g., model verification, reuse) after model development.

To sum up, there is a lack of a uniform method to formally represent and integrate domain knowledge with computational models in the current literature.

2.2 Formalizing Domain Knowledge to Support Lifecycle Activities of Computational Models

In this section, literature about formalizing domain knowledge to support lifecycle activities

of computational models has been reviewed. This section has been divided into three sub-sections that are related to the development, deployment and combination, and retrieval of computational models.

2.2.1 Formalizing Knowledge for Computational Models' Development

Recently, a lot of research has been carried out in bridging the semantic gap between data and computational models. For example, Johnson et al. (2010) proposed using an ontology to capture the domain concepts which are used to represent important variables for learning a decision tree. In the learning process, the decision tree model was iteratively learned from data and examined by domain experts. If the domain experts were not satisfied with the accuracy of classification, suggestions like adding variables to the model, merging variables into a new one, etc. were made. After that, the corresponding concepts were added to the ontology. Then the updated ontology was used to guide data processing. Although this was a good attempt in using ontologies to capture domain concepts for learning a decision tree, this study did not formally formulate the rules for data processing in the ontology. It also did not explicitly represent the decision tree and semantically connect the decision tree to the ontology. There is also similar research on formulating domain knowledge to bridge semantic gaps by Perez-Rey et al. (2006), Sinha and Zhao (2008), and Munger et al. (2015), etc. They also have problems in formally representing rules and integrating knowledge with analytic models.

There are also studies on using domain knowledge to construct analytic models. Campos and Castellano (2007) proposed learning a Bayesian Network structure by specifying the structural

restrictions from expert knowledge. The structural restrictions were defined as the existence of arcs, the absence of arcs, and causal ordering restrictions. These restrictions were claimed to be very useful in codifying the domain knowledge of a given domain. However, no specific domain knowledge formalization and integration were shown in this research. Lechevalier et al. (2016) introduced a domain-specific modeling approach to integrate a manufacturing system model with data analytics to facilitate effective and efficient data analytics in manufacturing systems. In their approach, the manufacturing meta-models, which define the concepts, rules, and constraints, are first captured. Taking the manufacturing meta-models and data as inputs, a Neural Network model builder computes the optimal number of hidden neurons and builds the optimal structure of the neural network. The generated Neural Network structure was recorded using a Neural Network meta-model. This meta-model was trained to obtain the final Neural Network model. In this research, although the manufacturing domain knowledge was captured, and the knowledge was used in creating a Neural Network structure, the domain knowledge and the Neural Network model's structure were loosely coupled. There were no mappings between the pieces of knowledge used for creating the structure and the specific structures (e.g., input neurons, hidden neurons, the structure of the neural network) that were captured explicitly and formally by the manufacturing meta-model. Again, the semantic links between the manufacturing meta-model and the Neural Network meta-model were missing. Kalet et al. (2017) proposed using a dependency-layered ontology, which was implemented in OWL, to solve the inconsistency and incompatibility between different Bayesian Network models in the medical domain. They utilized software tools to extract concepts that were related by a certain object property from the

developed ontology. Then, they applied software to automatically generate Bayesian Network topologies (i.e. nodes and arcs) based on the extracted concepts and relationships. However, the Bayesian Network model was not formally represented. Also, there were no semantic links between the developed Bayesian Network model and the ontology.

Hartmann et al. (2017) presented a model-driven analytics idea to emphasize the importance of using properly formulated domain knowledge in data analytics. They proposed to use a domain model to explicitly define the semantics of raw data in the form of metadata, domain formula, mathematical models, and learning rules. The domain model is used to guide the continuous refinement of the raw data to build a knowledge base. This knowledge base consequently contains the insight of the application domain, and it can be used for other applications. The advantages of applying the model-driven analytics are: (1) the modeled causal relationships within the data allow the refinement of only the necessary parts of data instead of recalculating everything; (2) it avoids the "store everything and analyze it later" strategy of today's pipeline-based analytics; (3) experts can describe their knowledge in the form of models, which enables what-if analysis; (4) the learning rules are organized with the domain data structure in a central place instead of being spread over the analytic tasks. However, the paper does not specify in what format to capture the metadata, mathematical formulas, and learning rules as well as how to integrate them. Also, the metadata model was not semantically connected to the analytic model.

To sum up, there are research gaps in (1) properly formulating domain knowledge for the development of computational models; and (2) integrating the formulated domain knowledge

with formally represented computational models to assist the construction of computational models. Thus, a modeling framework which formalizes domain knowledge and integrates the formalized knowledge and standardized computational models needs to be developed.

2.2.2 Formalizing Knowledge for Computational Models' Deployment

Research has also been carried out to formalize knowledge to support the deployment of computational models. The deployment of computational models in SM is a task that integrates one or more computational models in a Smart Manufacturing system. This requires the input and output data of computational models to be smoothly connected to the underlying system. Also, to deploy more than one computational model for an SM application, a methodology to combine the computational models needs to be developed. In this section, the literature that studies model deployment and model combination are reviewed.

Industrial efforts have been made to support model combination and deployment. Pivarski et al. (2016) introduced a new language, called the Portable Format for Analytics (PFA), for deploying analytic models into products, services, and operational systems. The PFA, which is currently under development by the Data Mining Group (DMG), is a JSON-based language that formally captures the analytic models along with the formalized functionalities of model consumption like data transformation and data aggregation. In their paper, it is emphasized that to use a programming language like JSON is safer than to use conventional programming languages like C, Python, or Java. This is because those conventional programming languages could access the underlying file system, operating system or network, which is not safe. The PFA

is expected to become a good tool to facilitate the deployment of analytic models. However, the PFA only supports analytic models. The deployment of other models is not supported. Also, the PFA does not provide a uniform way to model knowledge to support model deployment.

Another industrial effort that supports the interoperability of analytic models is the PMML (DMG, 2016), which is also developed by the DMG. The PMML is an XML-based language that enables the interoperability of analytic models. Currently, a lot of open source and commercial software tools support PMML. In PMML, a model combination method is included. In this method, several combination functions can be used like “majorityVote”, “average”, “max”, “selectFirst”, and “modelChain”, etc. Although some combinations of analytic models can be achieved using this method, a general approach to model the combination of all types of computational models is lacking. Moreover, the PMML can only capture the analytic model, and it lacks a mechanism to integrate domain knowledge with the analytic models.

There are also academic efforts towards formalizing domain knowledge and computational models for model deployment. For example, Brodsky et al. (2016) proposed a Sustainable Process Analytics Formalism (SPAF) to allow the formal modeling of modular, extensible, and reusable manufacturing process components with sustainability performance evaluation using mathematical programming-based optimization. The Optimization Programming Language (OPL) was used as the modeling language to build individual manufacturing process models and composite process models. Optimization models can be constructed based on the formalized query in OPL and can be further executed in IBM CPLEX. Although both manufacturing knowledge and the optimization model were formally represented and integrated by OPL, the

utilization of a specific modeling language (i.e. OPL) limits the reusability of such a model in software tools other than IBM CPLEX. This raises barriers to applying this approach in the SM systems which use assorted commercial and open-source software tools.

Kulkarni et al. (2016) proposed using a domain-specific modeling language to address difficulties in model composition in multi-discipline engineering analysis. The proposed domain-specific model language, which was named Model Composition and Analysis (MOCA), formally captured the entities and relationships of a domain. The general MOCA model has entities like Assembly, Component, Driver, DriverInterface, and DataPort. These entities can be used to represent engineering model compositions. The authors used Generic Modeling Language (GME) to implement the MOCA, and they developed a code generator which can generate Python code to integrate the MOCA instance models into the OpenMDAO platform. The OpenMDAO is an open-source computing platform for system analysis and multidisciplinary optimization. However, the proposed domain-specific modeling language only targets model composition. There are more types of model combinations that are not covered by the proposed language. More importantly, the developed domain-specific modeling language is developed to fit the framework of the OpenMDAO platform. This greatly limits the capability of this proposed language to be used in other software tools.

Shao et al. (2016) presented a research report on implementing a new ISO 15746 standard for chemical process optimization. The ISO 15746 standard describes a data model that facilitates integration between the advanced process control tools and engineering optimization tools. This research explored a Tennessee-Eastman chemical process in implementing the data

model defined in the standard. The populated data model was mapped to a metamodel of optimization problems (Assouroko and Denno, 2016), which was further serialized in OPL code. As mentioned in their work, there is no generic OptimizationDefinitionType defined in the standard, even though three types of optimization (i.e. steady state optimization, dynamic optimization, and expert system optimization) were captured. To use this standard, an optimization model's structures needed to be provided based on each optimization tool. The problem of uniformly defining a standardized optimization model was not addressed in the standard. Moreover, the integration between optimization models and manufacturing system remains at level 2 (i.e. monitoring and control of the production process) and level 3 (i.e. workflow/recipe control in production) of the ANSI/ISA-95 enterprise architecture (ANSI/ISA, 2010). A general integration framework was not provided.

In summary, there is a lack of a general method to model and integrate knowledge with computational models to support model deployment. Also, a general way to deploy computational models in different manufacturing systems is lacking.

2.2.3 Formalizing Knowledge for Computational Models' Retrieval

Today, with the increasing complexity of industrial systems, researchers and industrial users do not want to build their computational models of industrial systems from scratch. An alternative approach is to seek for pieces of existing models in order to build their models and build complex systems by combining smaller sub-models (Henkel et al., 2010). To facilitate this model reuse, a model retrieval process that decides on potentially suitable models from a large

number of available computational models becomes an important activity.

Henkel et al. (2010) presented a model retrieval approach for computational models in the biological domain. To enable the model retrieval, computational models in the biological databases were first annotated with meta-information. MIRIAM (Minimum Information Required in the Annotation of a Model) meta-information (Le Novère et al., 2005) was used for the annotation. The MIRIAM encompasses general information about a model like the model's name, author, and publication reference, etc. Their method also enabled queries based on model context information like queries about species, compartment and reaction, etc. An example in the paper showed their method can successfully retrieve models based on provided meta-information like model constituents description, author, date, and reference publication. Schulz et al. (2011) extended this research to apply a new semantic similarity measure to support model retrieval. They implemented the semantic similarity-based model retrieval method in a BioModels Database. The retrieved models were ranked by similarity scores, where higher scores indicated the retrieved model were more similar. However, neither study provided validations to prove their approach could accurately retrieve computational models based on queries.

Hoehndorf et al. (2011) proposed a framework to annotate system biology models with biomedical ontologies. The proposed framework proved to possess the capability to support model retrievals. They developed software tools to automatically convert annotated system biology models into OWL. Then, queries that were formalized by formal semantics could be performed in protégé. Although their approach proved to be able to discover computational models with queries, their query-based approach could not find models that were similar to the

queries.

Szabo and Teo (2011) presented a model retrieval approach for component-based simulation model development. To enable the retrieval, they first introduced an ontology that captures the important concepts and relationships of the component-based simulation model. A matching index to quantify the semantic relevance of the candidate simulation models from model discovery was proposed for model retrieval. The matching index was developed based on three parameters: model attributes, model component attributes, and model behavior. Ad hoc matching index values had been defined based on different matching conditions of the three parameters between a query and a model. However, verification and validation were not provided in the paper. Furthermore, having the matching index defined in ad hoc fashion makes it very difficult to apply the approach to other applications.

Li et al. (2017) proposed an ontology-based data mining model management method, and their method supports model selection. A DMMM (Data Mining Model Management) ontology had been developed to capture all the important concepts and relationships for all phases of a data mining process from business understanding to model deployment. In the example provided in the paper, an ontology-based data mining retrieval mechanism was demonstrated. By implementing the DMMM ontology in protégé, a new class can be created to represent the query and axioms can be developed for the class. Models that match the query can be inferred as a type of the created class when the reasoning engine is operated. An alternative approach for model retrieval is through developing Semantic Web Rule Language (SWRL) rules. Their approach can only retrieve models that match exactly the given query. Fuzzy searches for similar models were

not enabled.

Kannan et al. (2014) proposed a semantic annotation-based approach to discover environmental analytical models. They carried out their approach in two phases. In the first phase, a semantic network (i.e. ontology) has been built to capture key concepts in the air pollution domain. All the available analytic models were proposed to be modeled using the traditional conventions of semantic web services (i.e. input, output, preconditions, and effects). In the second phase, the concepts captured in the ontology and the types of the analytical model's entities were connected through relationships defined in the ontology. Such annotation enables the retrieval of an analytical model based on domain requirements. Experiments had been carried out and about 100% correct models were successfully retrieved based on given queries. However, their approach only allows exact queries. Fuzzy searches that can find relevant or similar models are not enabled.

To sum up, an ontology-based annotation approach for the retrieval of computational models is popular. For some type of computational models, the retrieved models need to be further combined. However, the current literature does not provide a method to support model retrieval based on similarities. Fuzzy model search based on similarities is important because model users may not always know the exact model they want to use. Similarity-based fuzzy search can provide model users a list of models that are similar to the requirements given by the model users. Also, the similarity values can provide model users with information about how related a model is.

CHAPTER 3. Proposed Methodology for the Knowledge Enriched Computational Model

In this section, the Knowledge Enriched Computational Model to support lifecycle activities of computational models is proposed.

3.1 General Description of the Knowledge Enriched Computational Model

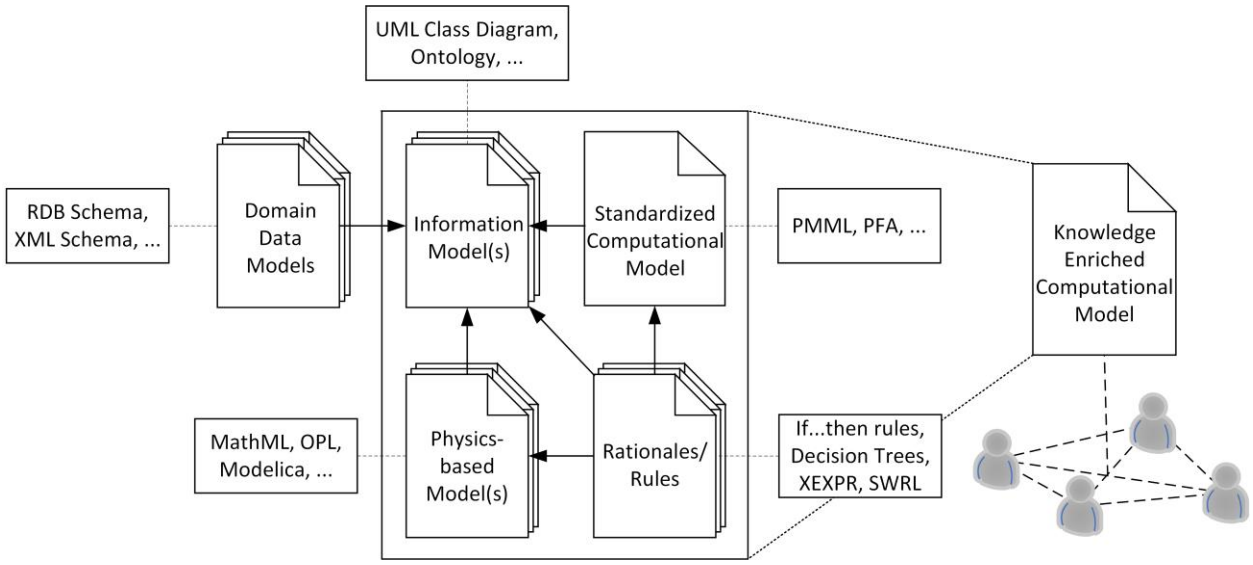


Figure 3.1 Enriching the standardized computational model with domain knowledge

To support the lifecycle activities of a computational model, knowledge that is needed from the SM domain can be: (1) the domain meanings of the computational model's entities (e.g., nodes, arcs, variables, etc.), (2) the physical or behavioral information that provides insights of a certain manufacturing system to which the computational model applies, and (3) descriptions or rules about how a computational model is developed, deployed or can be retrieved. To incorporate all these types of knowledge into a computational model and to enable the

interoperability and traceability of them, both knowledge and computational model should be explicitly and formally captured. In this dissertation, the three types of knowledge are captured into information model(s), physics-based model(s), and rationales, respectively. A standardized computational model should be used to formally capture the entities and the structure of a computational model. Figure 3.1 depicts the relationships between knowledge models and the standardized computational model.

To provide accessibility and availability of computational models and their relevant domain knowledge, the interoperability of this Knowledge Enriched Computational Model (KECM) must be enabled. In this dissertation, the text-based model interchange languages like XML, JSON, or OWL, etc. are used to formally represent the KECM. The advantages of using these languages are: (1) these text-based formats allow all software tools to parse, and (2) these languages cannot access the underlying manufacturing systems compared to C++, Java, and Python, etc. (Pivarski et al., 2016). All models (i.e. the standardized computational model, information model(s), physics-based model(s), and rationales) in the KECM should be represented in a single selected language (e.g., XML, JSON, or OWL, etc.). To support the traceability between different models within the KECM, the model entities which are related across models should be semantically connected.

The detailed descriptions of the four models are narrated in the following sections.

3.2 Information Model

In software engineering, an information model is a representation of concepts, relationships,

constraints, rules, and operations to specify data semantics for a chosen domain of discourse (Veryard, 1992). Here, the information model(s) provides a common terminology for the application domain where the computational model applies. Compared to the data models, which have implementation-specific details, information models define concepts and relationships in a higher abstract level and they are protocol neutral (Pras and Schoenwaelder, 2003). In the manufacturing domain, for example, the ANSI/ISA-95 standard is an information model that defines the concepts and relationships to support the interfacing between the enterprise business systems and the manufacturing control systems. The B2MML (Business To Manufacturing Markup Language) (Mesa International, 2013) is an XML-based data model which implements ANSI/ISA-95. Some examples of the other information models are: the MAnufacturing's Semantics ONtology (MASON) (Lemaignan et al., 2006), Manufacturing Reference Ontology (MRO) (Usman et al., 2013), Platform Independent Model (PIM) (Chungoora et al., 2013), and Process-oriented Information Model (PIM) (Zhang et al., 2015), etc.

Depending on the application domain, more than one information model may be needed if the computational model is developed for a cross-domain application. In these cases, the mappings between the information models should be explicitly defined to link concepts with the same meanings. It is important to notice that the information model(s) captured in the KECM should be the one(s) that are widely agreed to by the industrial community. This means that all companies accept and understand the information model(s) in the KECM. The utilization of an information model that is not agreed upon by the industrial community can bring difficulties in carrying out lifecycle activities of the computational model.

To explicitly express the domain meanings of other models (i.e. physics-based model(s), standardized computational model, and rationales), entities from other models need to be semantically linked to the corresponding concepts defined in the information model(s). Moreover, the information model(s) must possess the capability to be semantically connected to the domain data model(s).

3.3 Standardized Computational Model

The standardized computational model captured here is the formal representation of a computational model. Computational models like a rule model, an optimization model, a Bayesian Network model, etc. need to be formally represented. The entities in the computational model (e.g., nodes in a Bayesian Network) should be semantically connected to the corresponding domain concepts as defined in the information model(s). Since computational models can be developed for different domain applications, there are no semantic connections between the generally defined classes/types (e.g., node, arc, variable) in a computational model and the domain concepts (e.g., process, part, parameter, etc.). To enable semantic connections, both the standardized computational model and the information model(s) should be instantiated with respect to the domain application.

Currently, standards like MathML, PMML, and SMP2, etc. have already defined many standardized computational models. Although these standards capture computational models in a specific language like XML, their general model definitions (i.e. entity names and model structures) can be used to represent the computational model in other languages. The

standardized computational model needs to be represented in the same language as other models.

3.4 Physics-based Models

The physics-based models are the mathematical, empirical, simulation-based, and AI-based models, etc. that are developed to capture the physical mechanics of a phenomenon or the behaviors of an SM system. For example, forecasting models have been created to predict customer demand (Chapman, 2006) in the ERP level of a traditional hierarchical manufacturing system. At the MES level, production scheduling models have been studied for shop floor management (Pinedo, 2010). At the process level, cutting force models have been developed for modeling the material removal processes (Oberg et al., 2004). Though the models are developed for a certain manufacturing application, the physics/behavioral information in these models captures valuable insights about the manufacturing system.

Sometimes, physics-based models can only be processed by specific software tools. This is because these physics-based models are normally represented as application-specific languages. For example, the mathematical optimization problems can be modeled by the AMPL (A Mathematical Programming Language) and the OPL (Optimization Programming Language), which are processable in optimization solvers like CPLEX. To model complex systems in simulations, the object-oriented, declarative, multi-domain modeling language Modelica has been developed. Modelica can be processable by commercial or open-source tools like AMESim, Dymola, and Openmodelica. But it is very difficult to process the models outside these application-specific tools.

To enable a universal method to extract information from the physics-based models, these physics-based models need to be transformed into text-based formats (i.e. their standardized models). The text-based formats are friendly for software tools to parse. It should be noted that, no matter which text-based format (i.e. XML, JSON, or OWL, etc.) a physics-based model has, to merge a physics-based model(s) into a computational model, the physics-based model(s) should be finally transformed into the same format as the other models (i.e. information model(s), standardized computational model, and rationales).

3.5 Rationales

The rationales or rules are used to describe the rationality of a computational model or to guide the lifecycle activities of computational models.

For model development, rationales can be used to guide the development of computational models. For knowledge from the application domain, the rationales need to have connections to the related information models for obtaining the semantic meaning of the domain concepts. The rationales may also need to be linked to the physics-based models to indicate the part of system behavioral knowledge used in developing the computational model. Also, the rationales need to connect to the standardized computational model to specify the links between the computational model and the knowledge used in model development.

To facilitate the deployment of a computational model in a manufacturing system, the rationales capture rules that can load data between the information model(s) and the standardized computational model. The information model(s), which is accepted by the industrial community,

can be shared among different stakeholders without considering the specific data model(s). Although information models are normally used to capture higher-level information, they can also represent data when necessary.

To support the retrieval of computational models with similar functionalities, rationales capture the formal description of the computational model. In this dissertation, the formal description of a computational model is represented as formal semantic expressions. To retrieve models, a semantics-based method has been proposed to measure the semantic similarity between the semantic expression of a computational model and that of a model retrieval requirement. Only computational models with high similarity values can be retrieved.

Like other individual models in the KECM, the rationales also needed to be formally represented to make them processable and understandable by software tools. For rationales that are in rule-like fashion, some technologies that formally express rules can be used. For example, the XEXPR scripting language (W3C, 2000) enables the expression of rules in XML. JsonLogic (Wadhams, 2015) allows the construction of complex rules and serialization of the rules in JSON. In OWL, the SWRL (Semantic Web Rule Language) (W3C, 2004) language can be used to build rules. For rationales that are model descriptions, the native XML, JSON, or OWL languages can be used. The selection of the languages should conform to the overall representation technique.

CHAPTER 4. Utilization of the Knowledge Enriched Computational Model for Model Development

This chapter presents a method to use the KECM to support the development of computational models. A case study that develops a Bayesian Network to estimate energy consumption of the injection molding process has been introduced to demonstrate the utilization of the KECM. In the case study, the KECM for developing the Bayesian Network model has been developed. Finally, the benefits of using the KECM to support model development are discussed.

4.1 Introduction

Domain knowledge is normally heavily used in developing computational models. However, the knowledge is not properly captured and integrated with the standardized computational models. The model development knowledge is normally documented in natural languages with the computational models. However, no software tools can easily process and understand the documented natural language-based knowledge. Model development knowledge is important for downstream activities of computational models like model maintenance or model update. This is because whenever a model needs to be updated or modified, it is crucial to understand how the original model was developed. Domain knowledge brings the understanding of the domain meanings of the computational model's entities (e.g., nodes, arcs, variables, etc.) and means to construct the computational model. Moreover, for computational models that have to be developed in distributed environments, the interoperability of the domain knowledge used for model development should be enabled. For example, without explicitly and formally captured

model development knowledge, developing a data analytics model with domain experts and data analysts being at different locations relies solely on vocal discussions or written document exchange.

4.2 Development of Computational Models with the Knowledge Enriched Computational Model

The Knowledge Enriched Computational Model discussed in Chapter 3 possesses the capability to support the development of computational models. The KECM, which formally represents a computational model and its relevant domain knowledge, can be used as the medium for information exchange between model developers located in different geographical places (Figure 4.1). Model developers can directly work with the KECM to create or modify models and to add the corresponding knowledge (i.e. physics-based models and rationales). To test the model during development, parsers can be developed to transform the standardized computational models from the model interchange language like XML, JSON, and OWL to the means that software tools can consume.

To validate this, a case study has been created to utilize the proposed KECM to support the development of a Bayesian Network model. In the next section, a case study scenario of developing the Bayesian Network model without using the KECM is first introduced. Then, the utilization of KECM to support the development of the Bayesian Network model is described.

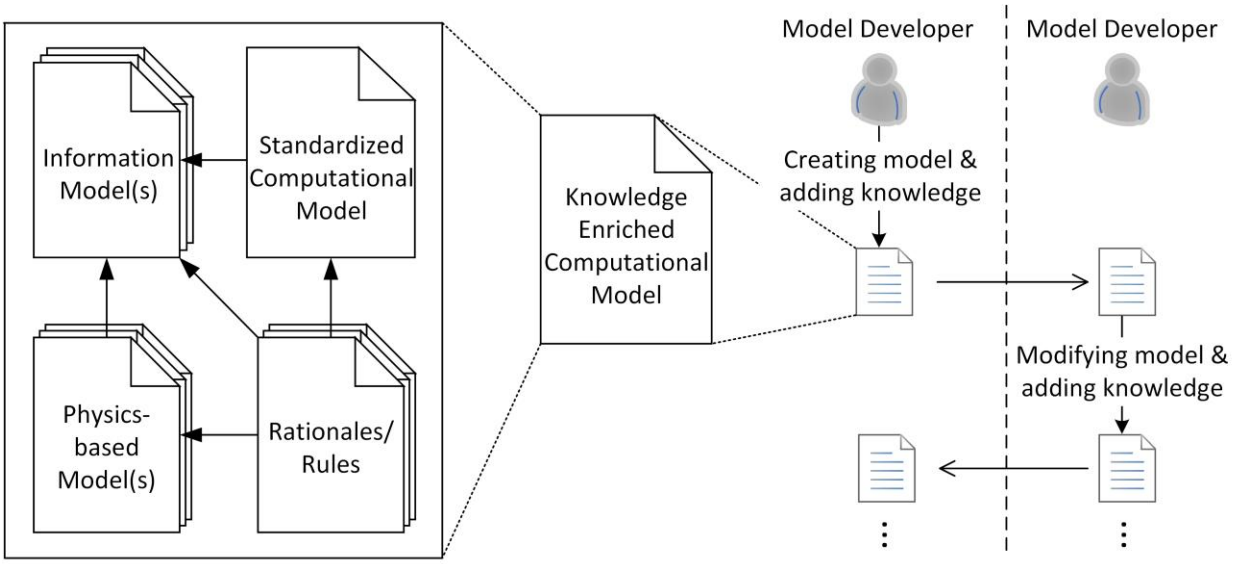


Figure 4.1 Development of computational models with the KECM

4.3 Case Study Scenario

In a previous study (Li et al., 2017), a Bayesian network (BN) model has been developed to predict the energy consumption of the injection molding process. The advantages of using a Bayesian Network to predict the energy consumption of injection molding are: (1) BN is suitable for small data sets. To train a BN model for energy estimation, data from part design, mold design, material, and machine needs to be available. Although injection molding is one of the mass-production processes, the collected data targeting at different products/parts may be limited. (2) A BN allows efficient use of different sources of knowledge: knowledge provided by domain experts and the knowledge learned from data. The ability to learn a BN structure from data can help the user to identify new relationships between parameters, which in turn can be used for process improvement. (3) A BN can answer queries based on incomplete information. A designer may not possess all the information like the properties of the injection molding machine that will

be used for producing the part. A BN can provide an estimate for a query considering nearly all possible values for that missing information based on the knowledge learned from data.

Table 4.1 Parameters for modeling the Bayesian Network nodes

Category	***Name***	***Unit***	***Description***
Product	V_p	m^3	Volume of the part
	Δ	N/A	Percentage of volume used for gating system
	d	mm	Maximum depth of the part
	n	N/A	Number of cavities
	h_m	mm	Maximum wall thickness
Material	$Materic$	N/A	Material type for the injection molded material
	ρ	kg/m^3	Specific density of polymer
	γ	$(mm^2)/s$	Thermal diffusivity of the material
	C_p	J/kg℃	Heat capacity of the polymer
	H_f	kJ/kg	Heat of fusion
	ϵ	N/A	Percentage shrinkage rate of the polymer
Machine	$Machin$	N/A	Machine type for the injection molding machine
	P_b	kW	Power consumption when the machine is idling
	s	mm	Maximum clamp stroke
	t_d	s	Dry cycle time
	P_{inj}	kW	Machine injection power
Process	pi	MPa	Injection pressure
	T_m	℃	Recommended mold temperature
	T_{inj}	℃	Injection temperature
	T_{ej}	℃	Ejection temperature
Environment	T_{pol}	℃	Initial temperature of the polymer
Others	Q	m^3	Maximum flow rate for injection
	Q_{avg}	m^3	Average flow rate
	P_m	kW	Melting power
	V_s	m^3	Volume of one shot including gating system
	E_m	kJ	Energy consumption in melting
	E_{inj}	kJ	Energy consumption of injection
	t_{inj}	s	Injection time
	E_c	kJ	Energy consumption in cooling
	COP	N/A	Coefficient of performance
	E_r	kJ	Energy consumption in resetting
	t_r	s	Resetting time
	E_{shot}	kJ	Energy consumption of a shot
	η	N/A	Efficiency
	t_{cyc}	s	Cycle time
	E_{part}	kJ	Energy consumption of a part

To study the role of SM domain knowledge in developing the BN, a BN model was first created by learning its structure and parameters from the data using the 'bnlearn' package (Scutari and Denis, 2014) in R without the intervention of the domain knowledge. The BN nodes were selected from the parameters related to the product, material, machine, process, and environment, etc. (Table 4.1). The parameters were extracted from Nannapaneni et al. (2016). After the learning process, the prediction accuracy was tested. It was achieved at 76.8%, which is relatively low for effective prediction. By carefully studying the structure of the learned BN (Figure 4.2), we found that the learned structure missed finding important relationships and captured wrong/weak relationships instead. To improve the learned model, expert knowledge was applied to identify the problems in the model. The BN development process is shown in Figure 4.3.

Due to the lack of LCA (Life-cycle assessment) data from the real injection molding processes, a simulation-based data generator had been developed to generate the data. This data generator has been validated against experimental data from the literature (Ribeiro, 2012). Before learning the structure from data, a whitelist which captures important relationships between the parameters was created. A whitelist, which contains arcs (that need to be included in the BN) was created based on the knowledge found from mathematical equations (shown in Table 4.2) for calculating the energy consumption of the injection molding process. The equations are extracted from Madan et al. (2013). An equation can be considered as defining the parent/child relationships for the equation variables. The independent variables (i.e. variables on the right-hand side of an equation) of an equation are treated as the parent of the dependent

variable (i.e. the variable on the left-hand side of an equation).

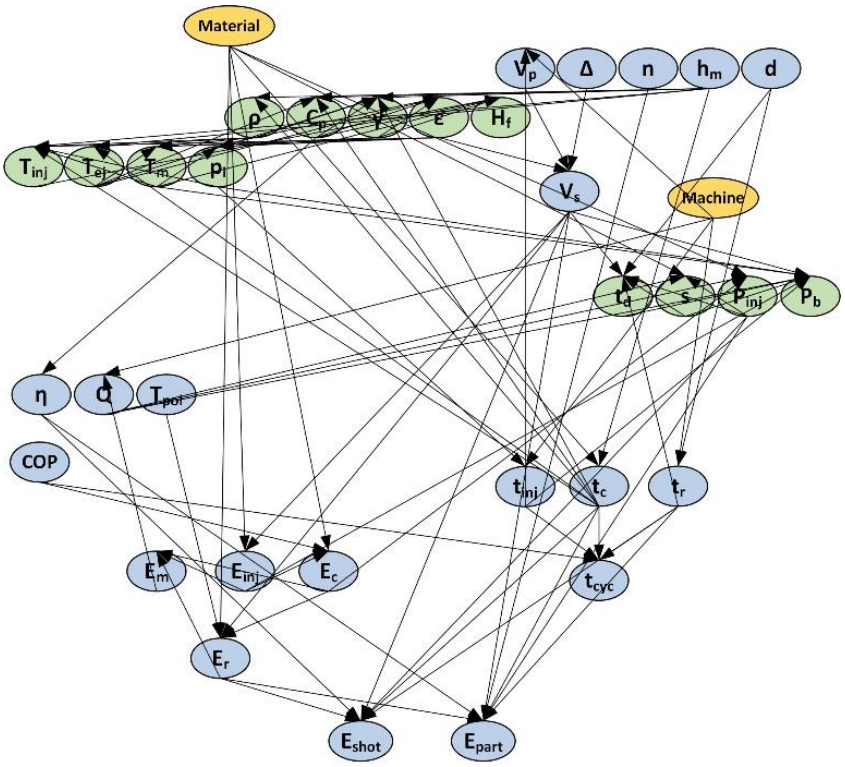


Figure 4.2 A BN structure learned from data

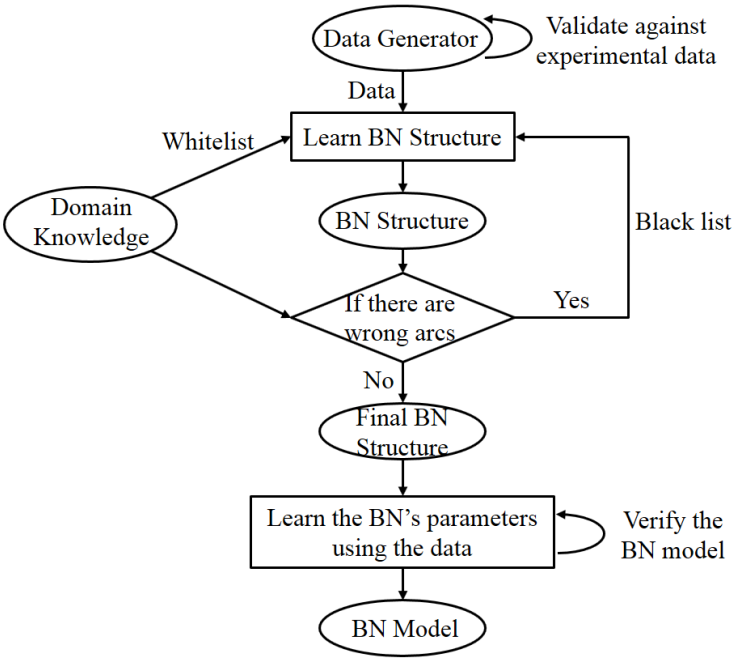


Figure 4.3 Development process for the BN

Additionally, a blacklist, which prevents the BN to create arcs between nodes, was created from the problems identified in the learned BN structure. Through carefully examining the learned BN structure (Figure 4.2), four problems were identified: (1) a parameter node (i.e. nodes defined by the *Parameter* class in the SMO) from one of the 5 categories (i.e. product, process, material, machine, and environment) should not have causal relationships with parameter nodes from the other 4 categories. For example, material-related parameters like ρ and C_p are found not to depend on the material but are related to a product-related property h_m. Though there are recommendations for the minimum wall thickness according to the injection molded materials, the maximum wall thickness h_m are normally designed as thinner as possible. This is because thinner walls require less material and less cooling time. However, there are no recommendations for the h_m given different materials. (2) The concept nodes like *Material* and *Machine* should not be related to the parameters from categories other than *Material* and *Machine*, respectively. Figure 4.2 shows that machine property nodes s and P_{inj} are found to be dependent on node *Material*. However, the injection molding machine is selected based on the shot size and the maximum clamp stroke, which are dependent on the product not material. (3) Parameter nodes within a category should not have parent-child relationships. It is true that within some categories like *Machine* and *Material*, parameter nodes are related. But, it is the material type or the machine type which determines the properties. (4) The parameter nodes from the 5 categories should not have any parent nodes other than the concept nodes. It can be observed in Figure 4.2 that parameter nodes from the 5 categories like T_{inj} and T_{ej} are found to have parent nodes in the *Others* category like t_{inj}. There may be causal relationships between T_{inj} and t_{inj}. But, it

should be t_{inj} to be dependent on T_{inj}, if there are causal relationships, not the other way around.

Table 4.2 Equations for estimating energy consumption of injection molding

Stage	***Equations***
Melting	$Q = P_{inj} * 1000/pi$
	$Q_{avg} = 0.5Q$
	$P_m = \frac{\rho Q_{avg} C_p (T_{inj} - T_{pol}) + \rho Q_{avg} H_f}{1000}$
	$V_s = V_p \left(1 + \frac{\epsilon}{100} + \frac{\Delta}{100}\right) n$
	$E_m = (P_m * V_s)/Q$
Injection	$E_{inj} = P_{inj} t_{inj}$
	$t_{inj} = \frac{2 V_s p_i}{P_{inj}}$
Cooling	$E_c = \frac{\rho V_s C_p (T_{inj} - T_{ej}) + \rho V_s H_f}{1000 \times COP}$
	$t_c = \frac{h_m^2}{\pi^2 \gamma} \frac{4(T_{inj} - T_m)}{T_{ej} - T_m}$
Resetting	$E_r = 0.25(E_{inj} + E_c + E_m)$
	$t_r = 1 + 1.75 t_d \sqrt{\frac{2d + 5}{s}}$
Whole Process	$E_{shot} = 1.2 \times \left(\frac{0.75 E_m + E_{inj}}{\eta_{inj}} + \frac{E_r}{\eta_r} + \frac{E_c}{\eta_c} + \frac{0.25 E_m}{\eta_m}\right) + P_b t_{cyc}$
	$t_{cyc} = t_{inj} + t_c + t_r$
	$E_{part} = \frac{E_{shot}}{n}$

By utilizing the whitelist and the blacklist, an iterative approach to learn the BN structure from data was applied (Figure 4.3). By using the iterative approach, the wrong arcs can be easily identified and handled during each iteration. With the whitelist and the iteratively updated blacklist, the learning procedure is repeated until no wrong arcs can be found in the BN structure. After learning the BN parameters (i.e. conditional probability tables for discrete nodes and Gaussian distributions for continuous nodes) and verifying the BN model, the development of the BN is finalized. The prediction accuracy of the BN model developed with the domain knowledge is achieved at 85%, which is satisfied and is higher than the learned BN model.

4.4 Development of the Knowledge Enriched Computational Model

In this section, the KECM for the BN is developed. The development of each individual model and the integration between the models are introduced. In this paper, OWL 2 (W3C, 2012) is used as the format for implementing all the models.

4.4.1 Information Model

As previously discussed, there are a lot of information models or ontologies developed in the manufacturing domain like MASON and MRO. Since the application domain of this case study is targeting estimating the energy consumption of the injection molding process, the information model used in this paper is selected from a previous work (Zhang et al. 2015). This information model was developed to facilitate the sustainability evaluation in the manufacturing domain. This model was also extended with respect to the injection molding process. A compact version of the information model, or the Sustainable Manufacturing Ontology (SMO), is extended for this case

study (Figure 4.4). A brief explanation of the concepts in the information model is narrated below:

- **Product**: A *Product* describes an object which is synthesized by a set of parts or subassemblies (each subassembly itself is also a product). The spatial relationships and contact constraints between parts are also defined within the *Product* class.
- **Part**: A *Part* is a single component that is used to construct a *Product*. A *Part* is a minimal functional unit of a product; thereby a part must be formed with a type of material and it has a certain geometrical shape.
- **Material**: A *Material* describes a kind of material associated with a *Part*. A *Material* has a list of properties like mechanical properties, chemical properties, thermal properties, etc. which are captured in the *Parameter* class.
- **State**: The *State* class describes the status of a *Product*, *Part* or *Material* at a certain time point. For example, a mechanical or a chemical property of a particular *Material* might have different values under different conditions or by using different measuring methods. The *State* class enables the SMO to capture the characteristics of any *Product*, *Part* or *Material* at any important time point.
- **Process Plan**: A *ProcessPlan* defines a sequence of manufacturing operations to produce a *Part*. The types of processes, types of equipment and operation parameters are specified in a *ProcessPlan*.
- **Process**: A *Process* describes a series of operations that need to be carried out to produce the final product. A *Process* can be a *ManufacturingProcess* or an *AssemblyProcess*. A

ManufacturingProcess is a process that transforms a raw material into a finished or a semi-finished *Part*. It can be a machining process, a casting process, a forging process, or a heat treatment process, etc. All the *ManufacturingProcesses* required to be carried out to produce a *Part* construct a *ProcessPlan*.

- **Activity**: An *Activity* is a minimal operational unit of a *Process*. For example, an *Activity* of a typical machining process can be setting up the machine, fastening the workpiece, positioning the cutting tool, injecting the cutting fluid, etc.
- **Environment**: The *Environment* class describes the environment related concepts of an *Activity* or a *Process*. All types of the environmental impacts are defined here, and each type of impacts is represented as a sustainability indicator. The sustainability of a *Part* or a *Product* can be further evaluated by considering the *Process*es that are carried out to produce the *Part* or *Product*.
- **SustainabilityMetrics**: The *SustainabilityMetrics* class is able to describe any of the sustainability metrics published in the literature or applied in the industry. As previously discussed, sustainability metrics are associated with their own evaluation methods (e.g., analytical models) and particular manufacturing processes. Thus, the *SustainbilityMetrics* can be attached to a certain *Process* or an *Activity*.
- **Parameter**: A *Parameter* represents an entity that describes a property of a manufacturing concept. The properties of a *Product*, a *Part*, a *Material*, a *Process*, and an *Activity* are modeled as *Parameter*s.
- **Equipment**: *Equipment* can be tools or machines on the shop floor.

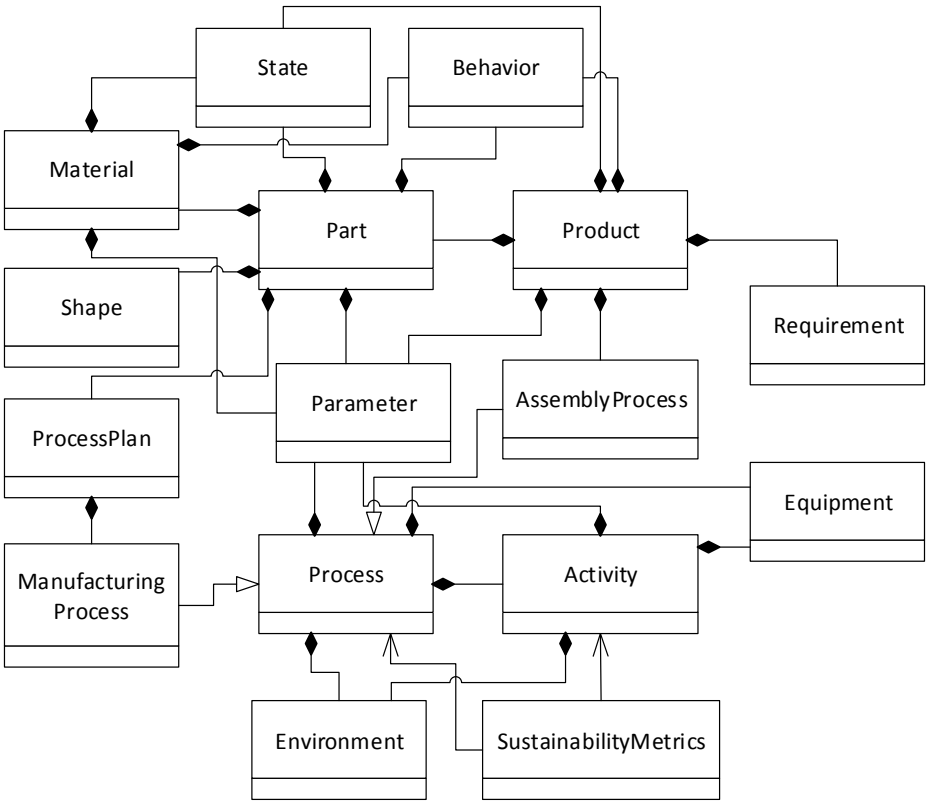


Figure 4.4 A UML Representation of the extended Sustainable Manufacturing Ontology (SMO)

4.4.2 Physics-based Model

The physics-based models used in developing the BN are the mathematical equations which estimate energy consumption of injection molding (Table 4.2). To represent mathematical equations in OWL, the OntoModel proposed by Suresh et al. (2008) has been used. In OntoModel (Figure 4.5), other than capturing the assumption, universal constant and dependent variable, etc., an equation is represented using the Content ML in MathML. In Figure 4.5, the black boxes represent *owl:class*es; the green boxes are datatypes; the pink arrows indicate the *hasSubClass* relationships; the red arrows indicate the *has-a* object properties; the green arrows indicate the data properties. The OntoModel is modified so that it can be connected to the

domain information model (i.e. SMO) as shown in Figure 4.4. The *Variable* class in the OntoModel is connected to the *Parameter* class in the SMO, which connects the variables in an equation to their domain meanings.

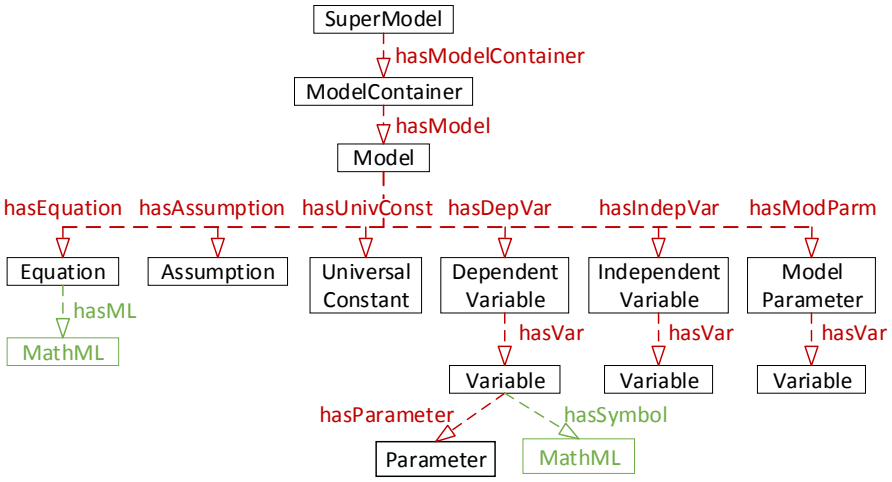


Figure 4.5 OntoModel and its Connection to the SMO

4.4.3 Standardized Computational Model

To fully represent a BN (i.e. the standardized computational model) in an OWL ontology, an OWL-based BN model is developed. Figure 4.6 demonstrates this OWL-based BN model in a tree structure. The class names in this model are borrowed from the PMML 4.3 - Bayesian Network Models (DMG, 2016). The structure of the PMML BN model is slightly modified (e.g., adding *BayesianNetworkNode* class, replacing the *has-a* relationship between ContinuousDistribution and *NormalDistribution* with the *hasSubClass* relationship) to better fit the OWL structure. This OWL-based BN model has been verified with the BN example provided on the webpage of the PMML BN model. The verification proves the OWL-based BN model to be capable of fully representing BNs.

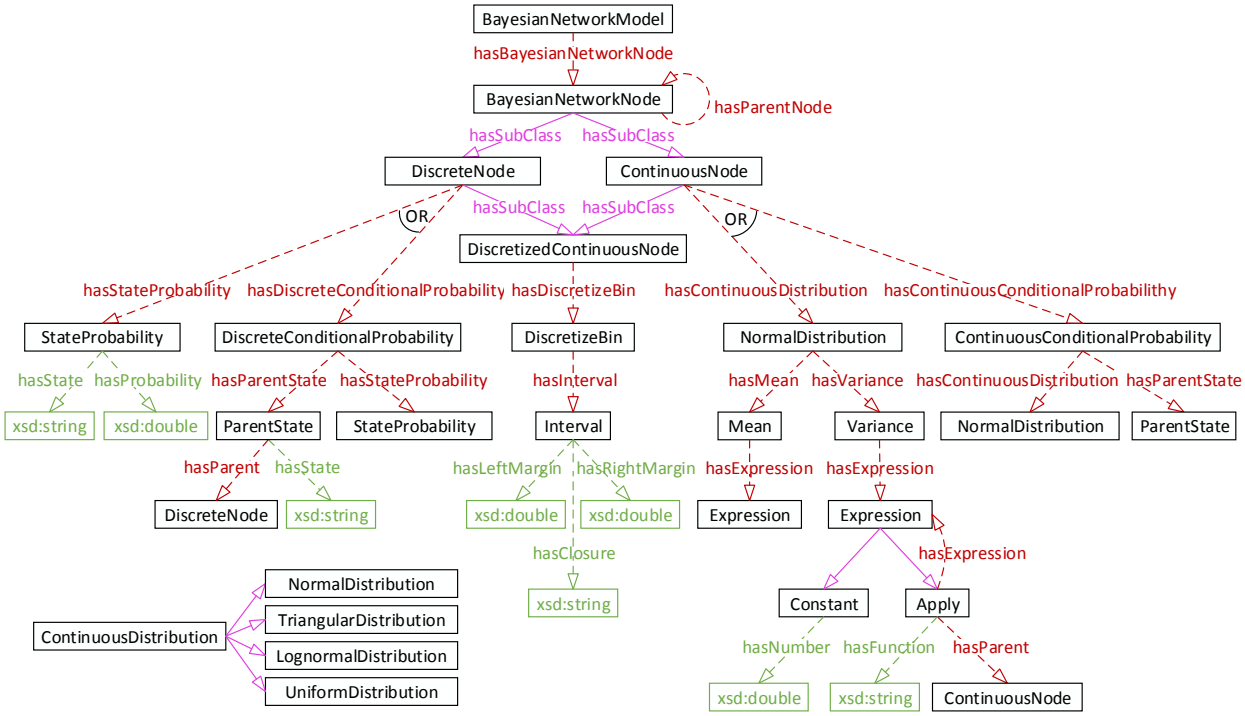


Figure 4.6 A Tree presentation of the OWL-based BN model

All the parameters in Table 4.1 are modeled as the *BayesianNetworkNode* instances in the OWL-based BN model. The semantic connection between a *BayesianNetworkNode* and a manufacturing concept in the SMO is achieved by an isAssociateTo(*BayesianNetworkNode, domainConcept*) object property.

4.4.4 Rationales

To improve the BN structure with domain knowledge, the rationales/rules to facilitate the creation of the whitelist and the blacklist are developed. The whitelist rules/rationales are created to capture the BN node relationships provided from the physics-based models (i.e. equations) and domain rules. Based on the identified four problems of the learned BN structure (section 4.3), blacklist rules/rationales are developed. The blacklist rules can be used to avoid the wrong

structures in the BN. All the whitelist and blacklist rules are modeled using SWRL in OWL. The *hasParentNode* object properties represent the whitelist relationships. The *hasWrongArc* object properties represent the blacklist relationships. To enhance the traceability of the rules, each rule has its own corresponding numbered object property. For example, the *hasParentNode* object property in whitelist rule 1 is *hasParentNode1*.

Whitelist Rule 1

This rule is created based on the physics-based models (equations in Table 4.2). As discussed before, an equation can be considered as defining the parent/child relationships for the equation variables. The independent variables (i.e. variables on the right-hand side of an equation) of an equation are treated as the parents of the dependent variable (i.e. the variable on the left-hand side of an equation). The *hasParentNode1* object property represents the parent/child relationship between two *BayesienNetworkNode*.

DependentVariable(?dv), IndependentVariable(?iv), MathematicModel(?m), Variable(?v_dv), Variable(?v_iv), Parameter(?p1), Parameter(?p2), BayesianNetworkNode(?n1), BayesianNetworkNode(?n2), hasDependentVariable(?m, ?dv), hasIndependentVariable(?m, ?iv), hasVariable(?dv, ?v_dv), hasVariable(?iv, ?v_iv), isAssociatedWith(?n1, ?p1), isAssociatedWith(?n2, ?p2), isAssociatedWith(?v_dv, ?p1), isAssociatedWith(?v_iv, ?p2) -> hasParentNode1(?n1, ?n2)

The meaning of this rule is: for any *MathematicModel*, if the *Variable*s of its *IndependentVariable* and its *DependentVariable* represent the same *Parameter*s as two

*BayesianNetworkNode*s do, then the *BayesianNetworkNode* that can represent the *IndependentVariable* in the *MathematicModel* should be the parent node of the *BayesianNetworkNode* that represents the *DependentVariable* in the *MathematicModel*.

Whitelist Rule 2

According to the classification of the parameters in Table 4.1, the manufacturing concept nodes (e.g., *Machine* and *Process*) should have parent-child relationships with their related parameter nodes.

ManufacturingConcept(?mc), Parameter(?p), BayesianNetworkNode(?n1), BayesianNetworkNode(?n2), isAssociatedWith(?n1, ?mc), isAssociatedWith(?n2, ?p), hasParameter(?mc, ?p) -> hasParentNode2(?n2, ?n1)

The meaning of this rule is: for any *ManufacturingConcept*, which can be *Machine*, *Product*, and *Process, etc.* because they are *sub-class*es of *ManufacturingConcept*, its corresponding *BayesianNetworkNode* should be the parent node of the *BayesianNetworkNode* that represents the *Parameter* of the *ManufacturingConcept*.

Whitelist Rule 3

Some process parameters in the injection molding process are selected according to the material. For example, the selection of T_{inj}, T_{ej}, T_m, and p_i are selected based on the material type (Boothroyd et al., 2011). So, causal relationships between the *Material* node and these process parameters should be captured.

Material(?m), Parameter(?pp), Process(?p), BayesianNetworkNode(?n_m), BayesianNetworkNode(?n_pp), isAssociatedWith(?n_m, ?m), isAssociatedWith(?n_pp, ?pp), hasParameter(?p, ?pp) -> hasParentNode3(?n_pp, ?n_m)

The meaning of this rule is: the *BayesianNetworkNode* which represents a *Material* should be the parent node of the *BayesianNetworkNode* which represents a *Parameter* of a *process*.

Blacklist Rule 1

To address the first problem of the learned BN structure, a set of rules to prevent connecting parameter nodes from different categories are created. Here, the rule to prevent parameter nodes from the *Material* and *Product* categories to be connected is demonstrated.

Material(?material), Parameter(?p_material), Parameter(?p_product), Product(?product), BayesianNetworkNode(?n_p_material), BayesianNetworkNode(?n_p_product), isAssociatedWith(?n_p_material, ?p_material), isAssociatedWith(?n_p_product, ?p_product), hasParameter(?material, ?p_material), hasParameter(?product, ?p_product) -> hasWrongArc1(?n_p_material, ?n_p_product), hasWrongArc1(?n_p_product, ?n_p_material)

The meaning of this rule is: Any *BayesianNetworkNode* that represents a *Parameter* of *Material* should have *hasWrongArc1* relationships (i.e. two directions) with the *BayesianNetworkNode* that represents a *Parameter* of *Product*.

From this rule, it can be observed that the Blacklist Rule 1 tries to enumerate all the wrong arcs of problem #1.

Blacklist Rule 2

This blacklist rule addresses problem #2. It avoids the *Material* and the *Machine* nodes to be connected to the parameter nodes from other categories. An example rule is shown below to prevent the *Material* node to be connected to the machine-related parameter nodes.

Machine(?machine), Material(?material), Parameter(?p_machine), BayesianNetworkNode(?n_material), BayesianNetworkNode(?n_p_machine), isAssociatedWith(?n_material, ?material), isAssociatedWith(?n_p_machine, ?p_machine), hasParameter(?machine, ?p_machine) -> hasWrongArc2(?n_material, ?n_p_machine), hasWrongArc2(?n_p_machine, ?n_material)

The meaning of this rule is: Any *BayesianNetworkNode* that represents a *Parameter* of *Machine* should have *hasWrongArc2* relationships (i.e. two directions) with the *BayesianNetworkNode* that represents *Material*.

From this rule, it can be observed that the Blacklist Rule 2 tries to enumerate all the wrong arcs of problem #2.

Blacklist Rule 3

Targeting at problem #3, this blacklist rule avoids the parameter nodes within one category to be connected to each other. The example for the material category is shown below.

The meaning of this rule is: For any two different *BayesianNetworkNode*s that represent two *Parameter*s of *Material*, they should have the *hasWrongArc3* relationships (i.e. two directions).

From this rule, it can be observed that the Blacklist Rule 3 tries to enumerate all the wrong

arcs of problem #3.

Material(?m), Parameter(?p1), Parameter(?p2), BayesianNetworkNode(?n1), BayesianNetworkNode(?n2), isAssociatedWith(?n1, ?p1), isAssociatedWith(?n2, ?p2), hasParameter(?m, ?p1), hasParameter(?m, ?p2), DifferentFrom (?n1, ?n2) -> hasWrongArc3(?n1, ?n2)

Blacklist Rule 4

To prevent the parameter nodes from the 5 categories to have any parent nodes other than their corresponding concept nodes (problem #4), an example rule is demonstrated below for the process category. In this rule, the "*hasTempParentNode*" object property represents an arc learned from data.

Process(?process), Parameter(?pm), hasParameter(?process, ?pm), BayesianNetworkNode(?n_p_m), BayesianNetworkNode(?n_mc), isAssociatedWith(?n_p_m, ?pm), hasTempParentNode(?n_p_m, ?n_mc) -> hasWrongArc4(?n_p_m, ?n_mc)

The meaning of this rule is: If the *BayesianNetworkNode* that represents a *Parameter* of *Process* has a temporary parent node (i.e. *hasTempParentNode*) with any *BayesianNetworkNode*, then this arc (i.e. *hasTempParentNode* relationship) should be categorized as wrong arc type 4. Since the *hasTempParentNode* is the arc learned from data, so the relationship between the *BayesianNetworkNode* of a *Parameter* of *Process* and that of *Process* cannot be picked up again

by the learning algorithm (i.e. the relationship has already captured in the whitelist).

4.4.5 Representation of the Knowledge Enriched BN Model in OWL

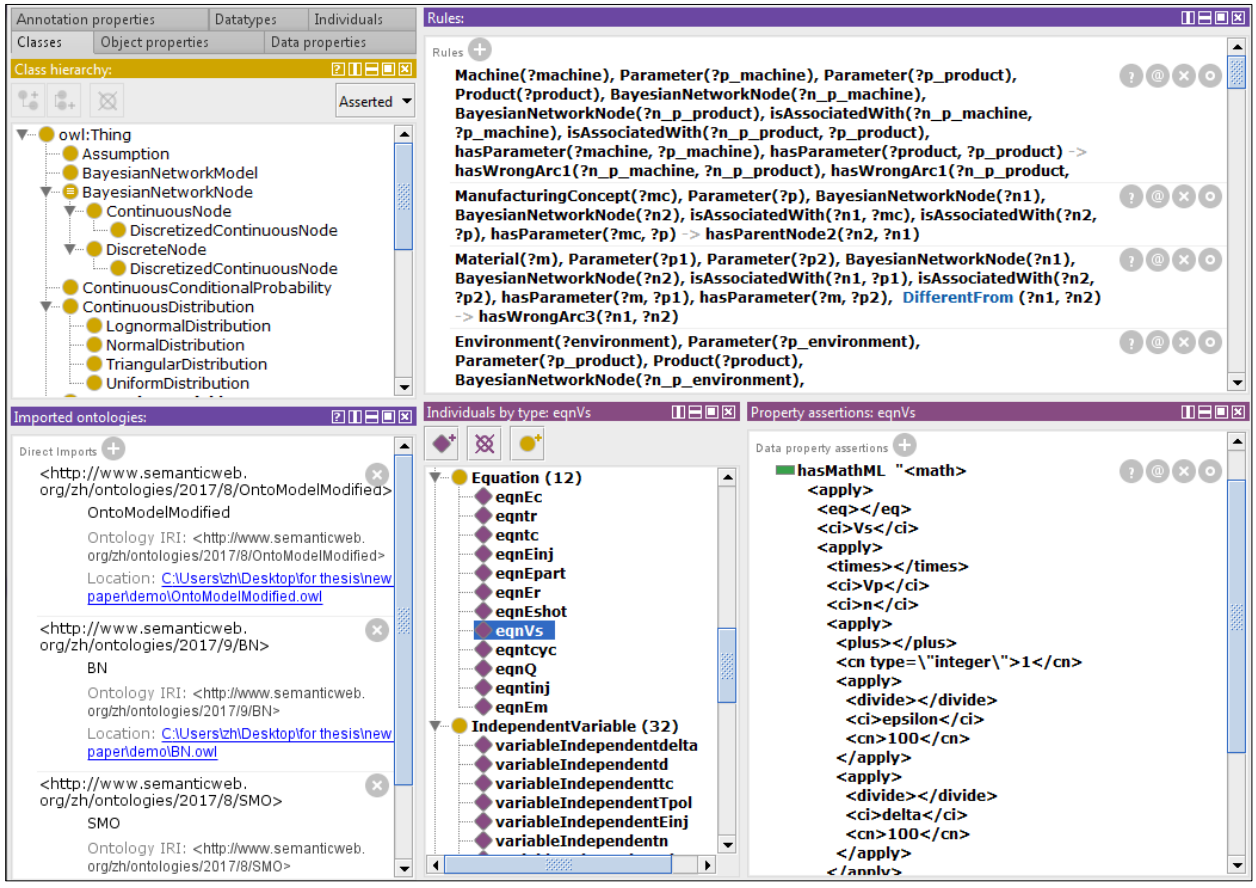


Figure 4.7 The Knowledge Enriched BN Model in protégé 5.2

In the development of the Knowledge Enriched BN model, the individual models for the information model, standardized computational model, and physics-based model are separately created first. These models are generic and could be applied to any applications and do not have any populated instances. After verifying that all the individual OWL-based model can sufficiently represent the models, the KECM is created by importing the three OWL-based models into the OWL-based KECM. Instances of the domain concepts in the SMO, the

BayesianNetworkNodes in the BN model, and the equations represented by the OntoModel are populated. The whitelist and blacklist rules are then created using the SWRL. The screenshot of the Knowledge Enriched BN model in protégé 5.2 is demonstrated in Figure 4.7.

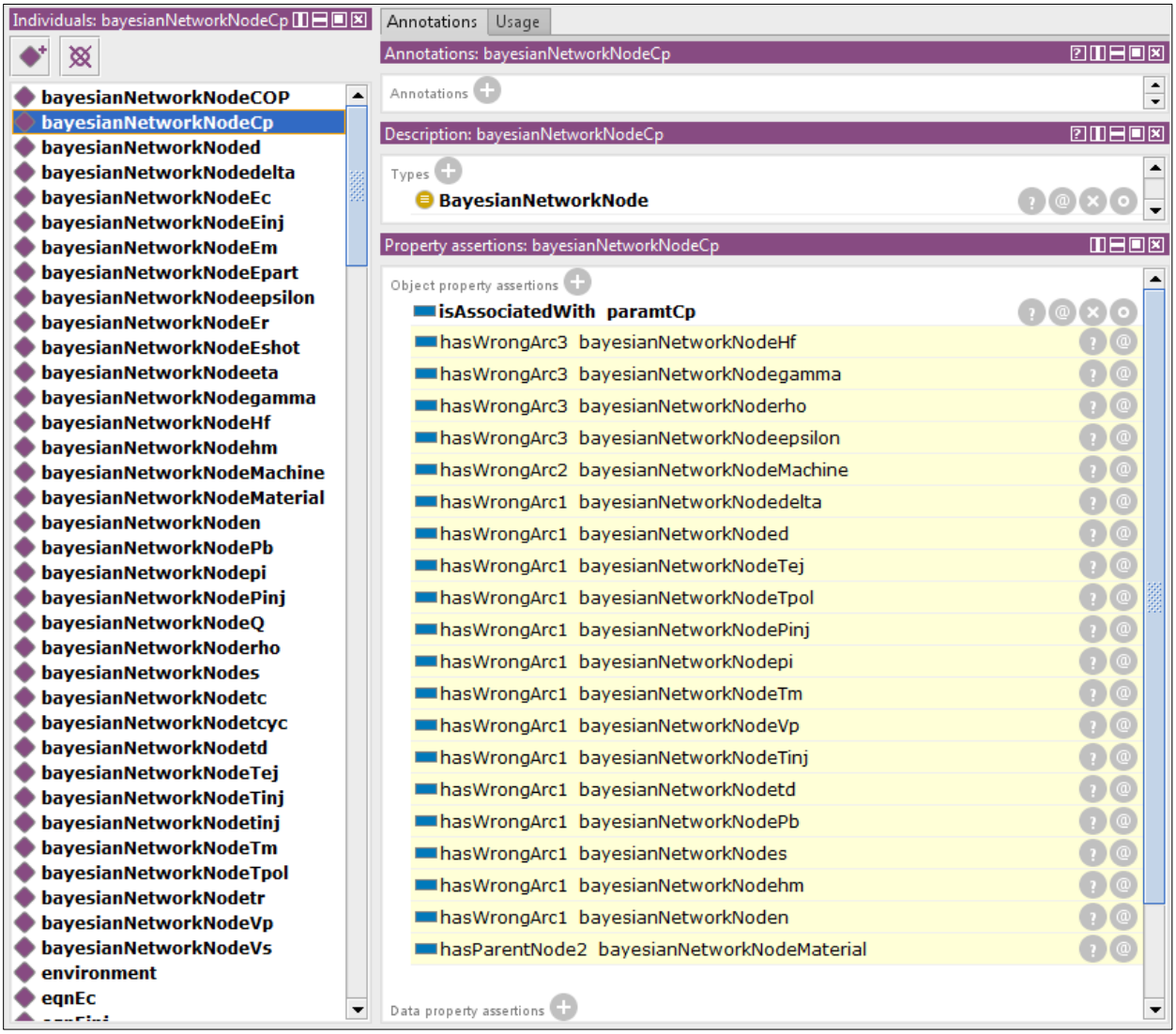


Figure 4.8 Inferred whitelist and blacklist relationships in protégé 5.2

Figure 4.8 shows a screenshot of the reasoning of rationales/rules (i.e. whitelist and blacklist rules). The object property assertions highlighted in light yellow are the inferred relationships

from reasoning the rationales/rules. It can be observed that the rationale/rule used to create or eliminate an arc can be easily tracked by using the numbered object properties.

4.5 Utilization of the Knowledge Enriched Computational Model

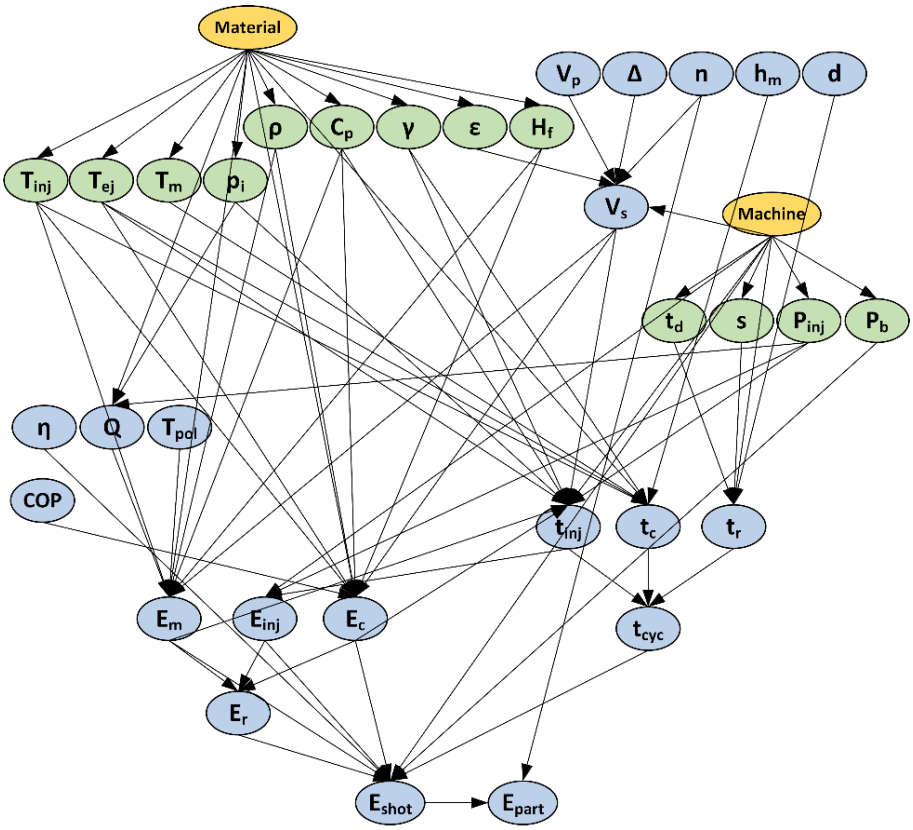


Figure 4.9 The final BN structure

Following the manual development process (Figure 4.3), a BN is developed (Figure 4.9) by utilizing the KECM. In the development process, the KECM has been used to exchange information between a domain expert and a data analyst. The domain expert first models the domain knowledge (integrating the information model, adding the physics-based model, creating rationales) for the development of the BN. Then, the data analyst iteratively learns the BN

structure from data with the whitelist and the blacklist, which are extracted from the Knowledge Enriched BN model through a parser (that has been developed for this purpose). Here, the KECM is used to pass the BN along with its relevant domain knowledge between the domain expert and the data analyst. After sending the KECM with the learned structures, the domain experts can analyze the BN structure and add the corresponding rationales to improve the BN structure. The domain expert can directly add or modify domain knowledge on the KECM in GUI (Graphical User Interface) -based software tools like protégé. Figure 4.10 is a sequence diagram that shows the information exchange.

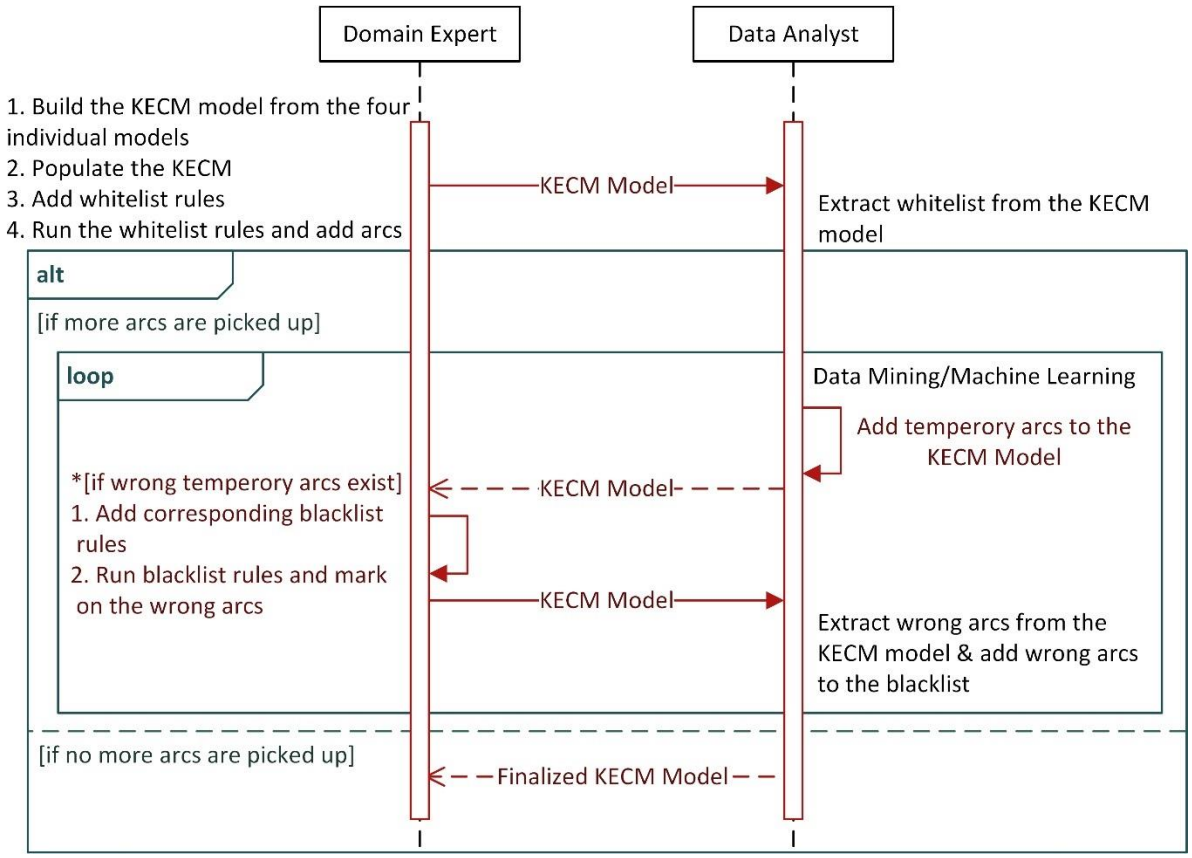


Figure 4.10 Information exchange using the KECM

4.6 Discussion

Two benefits have been identified during the development of the BN using the KECM: (1) Shortened cycle time for model development. In this case study, with the assistance of automatic processing and reasoning from the software tools, the development cycle time has been reduced from 2 hours to 15 minutes. With the formally defined rationales/rules, the arcs captured by the whitelist and the blacklist can be generated automatically instead of working manually. (2) Eliminating human error. Through reasoning, the formally defined rationales/rules, more arcs in the whitelist and the blacklist have been identified. Some of these arcs are missed by manual work.

These two direct benefits are brought by the enhanced interoperability and traceability of the KECM. From enhancing the interoperability perspective, the KECM can explicitly represent the computational models with their relevant domain knowledge through capturing their concepts, relationships, and rules, etc. without semantic ambiguity; the computational models and their relevant domain knowledge are formally represented, which enables the automatic processing through software tools. From enhancing the traceability perspective, the entities of a computational model can be directly traced to its corresponding domain concepts; the computational models' structures can be easily traced to the rationales/rules which create these structures. It can be expected that with the enhanced interoperability and traceability, more effective and efficient information exchange can be achieved by using the KECM in the distributed environment, where model developers are not in the same geographical area. The information exchange between them can be made explicit and formal by using the KECM

instead of online vocal discussion and written document exchange.

The limitation of using the KECM for model development is that formulating the physics-based models relies on their formal representations. In the case study, the representation of mathematical equations in the OntoModel is demonstrated. Enabled by the PMML and the PFA, formal representations of the predictive models can also be achieved. However, formal representations of other types of models like optimization models and simulation models are either limited or currently unavailable. Though the modeling languages for these models exist, these languages are tool-specific. Without the corresponding software tools and their APIs, the parsing of the programming language-based model is difficult. Thus, more work needs to be done for standardizing the physics-based models. This will also improve the interoperability of the physics-based models in their own domain applications.

CHAPTER 5. Utilization of the Knowledge Enriched Computational Model for Model Deployment

This chapter presents general methods to support the deployment of computational models. This chapter is divided into two main sections. The first section introduces a general method to deploy computational models in manufacturing systems. An example of deploying an optimization model in a B2MML-based system is illustrated. The second section describes a general method to represent the combination of computational models. A case study that demonstrates the combination of an Agent-based model and a Decision Tree model for a real-time scheduling application has been developed to illustrate the proposed method.

5.1 Utilization of the Knowledge Enriched Computational Model to Support the Deployment of Computational Models in Smart Manufacturing System

Currently, although the PFA has started to develop models to integrate standardized analytic models and methods to deploy them, there is a lack of general methodology to support the deployment of all types of computational models. However, to support the plug-and-play capability of computational models as required by Smart Manufacturing, a general methodology must be developed to integrate computational models with the necessary model deploying knowledge to allow different manufacturing systems to easily deploy the computational models. This knowledge includes the configuration of the model's parameters, preprocessing of the data to be consumed by the computational model, and storing the model results according to the SM system's underlying database or data exchange protocol, etc. In this dissertation, the data

integration between the computational models and the underlying SM system is focused on. The data integration knowledge is crucial in deploying computational models in the SM environment. For example, a computational model associated with a process control unit needs to be re-executed based on the real-time or near real-time data of process changes. Without smooth integration between the computational model and the SM system's underlying information system, the model solution may give wrong suggestions based on delayed data exchange. Without the formally captured and integrated knowledge to go with the computational model, no software tools can process and understand the model and its relevant knowledge. In this situation, the deployment of computational models relies heavily on manual work. The necessity of integrating model deployment knowledge into the standardized models comes from security reasons. A model consumer implemented in conventional programming languages (e.g., C, Java, or Python, etc.) could access the underlying file system, operating system, or network (Pivarski et al., 2016). But a model consumer that deploys models by consuming the standardized models can only transform the data that it is given. So, the security of an SM system that deploys a computational model can be enhanced if the deployment knowledge can be integrated with the standardized computational model.

In the following sub-sections, a model deployment methodology that conforms with the KECM has been presented. As a proof-of-concept, a case study has been presented to demonstrate the utilization of the proposed model deployment method to support the deployment of optimization models. Due to the lack of a standardized model for optimization models, an Optimization Metamodel that can formally represent optimization models has been developed.

5.1.1 A General Method to Support the Deployment of Computational Models

Based on the KECM, a general method has been developed (Figure 5.1) to support the deployment of computational models. In Figure 5.1, the information model(s) to be included in a KECM serves as a bridge to connect an SM system's data and the computational model's data. As stated previously, the information model(s) in the KECMs should be agreed upon by the industrial community. This means that all industrial users can understand and utilize the information model(s). To semantically connect the KECM to the SM system, the mappings between an industrial user's SM system's local data system (or its local data exchange protocol) and the information model(s) should be defined by the industrial user. The mappings between the standardized computational model and the information model(s) can be captured in the rules or the semantic links between model entities. Through the information model(s) and the mappings, the input/output data of a standardized computational model can be smoothly integrated with an SM system's data. The data from an SM system can first be loaded onto the information model(s) through mappings defined by the industrial users. Then, the data stored with the information model(s) can be loaded to the standardized computational model through the mappings between them. So, as long as the industrial users have defined the mappings between their SM system's data system and several industry-accepted information models, all standardized computational models that are integrated with these information models can be smoothly deployed in their SM systems.

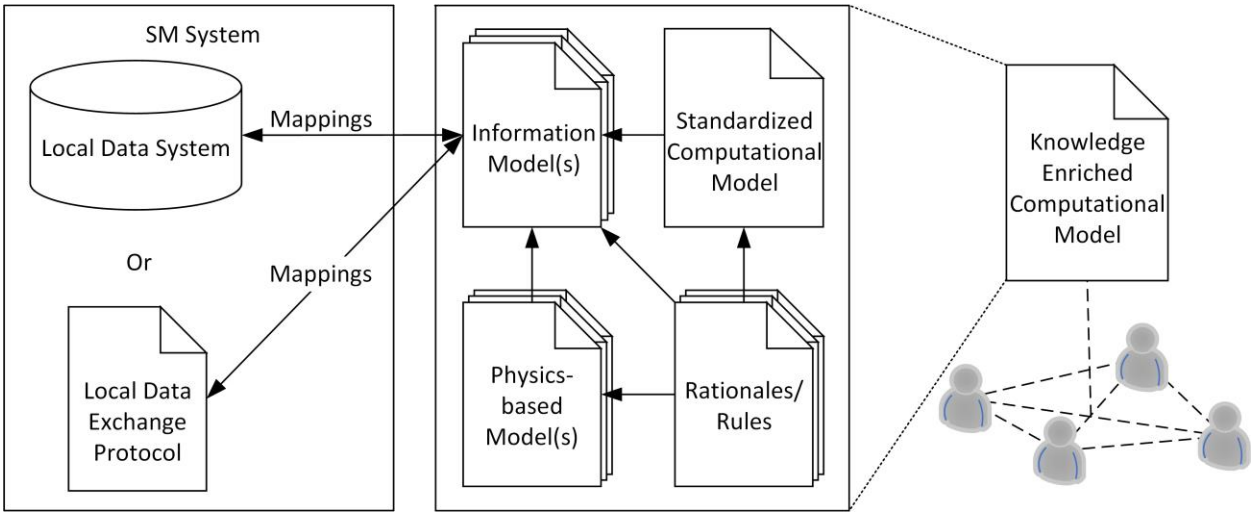


Figure 5.1 A general method to support the deployment of computational models

5.1.2 Development of A Standardized Model for Optimization Models

Among a variety of computational techniques, optimization plays an important role in both business and engineering decision making at all levels of an enterprise's hierarchy from business planning and logistics, manufacturing operations and control to batch and unit process control. However, due to the diversity of implementation environments of optimization applications, modeling languages for operation problems are often tool-specific. For example, OPL is used in the IBM CPLEX optimization solver; the AMPL supports more tools like the AMPL solver, Gurobi optimizer, and CPLEX, etc.; the GAMS (General Algebraic Modeling System) language can be connected to a group of optimization solvers like BARON, CONOPT, and CPLEX, etc. through the GAMS integrated development environment (IDE); MATLAB's Optimization Toolbox uses MATLAB's proprietary language; and the open source tool Google OR-Tools utilizes general-purpose programming languages like C++, Python, C#, and Java. Although efforts have been made to enhance the interoperability of some optimization modeling languages

like AMPL and GAMS, the interoperability is limited by the software tools supported by the modeling language. Developing, integrating and reusing optimization models still rely on the availability of specific software tools. This current situation hinders the accessibility and availability required by the SM systems. Difficulties in enabling the interoperability among these software tools result from information gaps in the inconsistent optimization terminologies, the large number of optimization methods (e.g., mathematical programming, constraint programming, and genetic algorithm etc.) that have been created, and the lack of communication standards between existing optimization tools (Witherell et al., 2007).

This section first introduces the development of an Optimization Metamodel to formally represent optimization models. Next, an example of a Mixed Integer Linear Programming (MILP) optimization model to solve a Flexible Job Shop Scheduling (FJSS) problem is described to demonstrate the representation of optimization models using the Optimization Metamodel.

5.1.2.1 Optimization Metamodel

The Optimization Metamodel is developed based on the compilation and organization of optimization terminologies gathered from the literature. To represent the Optimization Metamodel, machine-readable and understandable formats like XML, JSON, and OWL, etc. can be used. Figure 5.2 shows the Optimization Metamodel represented in a UML class diagram. In the UML notations, the black diamond arrows represent the “composition” relationships and the hollow arrows represent the “inheritance” relationships.

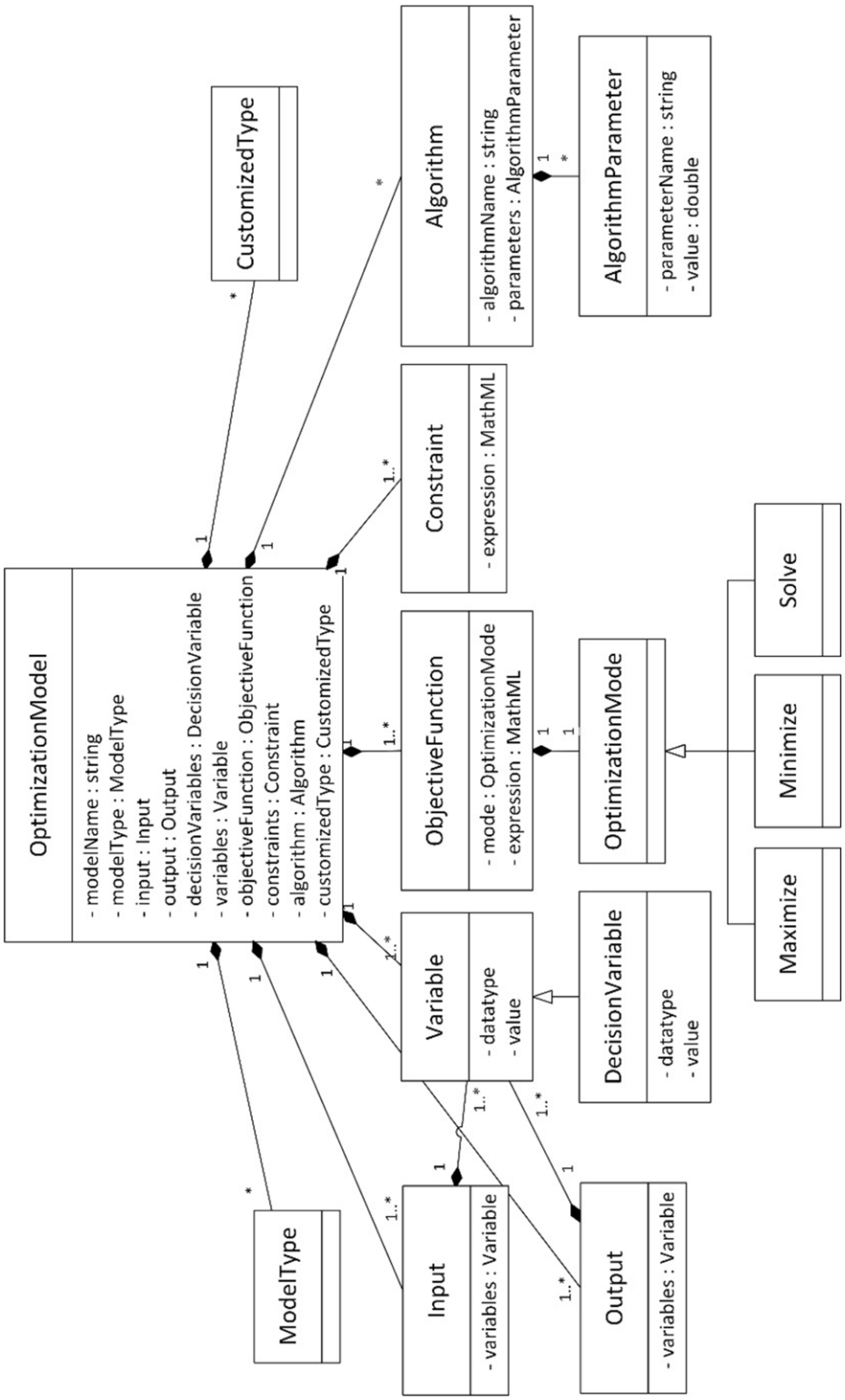


Figure 5.2 A UML representation of the Optimization Metamodel

In this model, the *OptimizationModel* class represents the highest-level entity of an optimization model. All the information about an optimization model should be captured within this class. To facilitate the definition of customized data types (i.e. data types other than double, integer, string, etc.), the *CustomizedType* can be used to represent a user-defined data structure. For example, an interval type, which can be used to solve scheduling optimization problems, can be defined as a *CustomizedType*. The *Variable* class captures variables that are used in the optimization model. A *DecisionVariable*, which needs to be determined to solve the optimization problem, is a sub-type of a *Variable*. The variables, which are determined by the system environment and have fixed values, should be modeled as *Variable*s. The *value* field of a *Variable* is used to contain the input value. The *value* field of a *DecisionVariable* is used for capturing the resultant value of a determined decision variable after an optimization model is resolved. The *Input* and *Output* classes indicate the list of input and output variables, respectively. The *ObjectiveFunction* class represents an objective function that is to be optimized. More than one *ObjectiveFunction* can be included in an *OptimizationModel* if the model is a multi-objective optimization model. An *ObjectiveFunction* has an *OptimizationMode* to indicate the means to achieve the objective function like *Maximize*, *Minimize*, or *Solve*. The *Solve* mode captured here is targeting some of the problems which only find feasible solutions instead of optimal solutions (e.g., Constraint satisfaction problems). The mathematical expression of the *ObjectiveFunction* is captured in the *expression* field. The constraints are represented by the *Constraint* class. The *Algorithm* class is used to indicate the specific algorithms to be used to resolve the optimization model. An *AlgorithmParameter* represents a parameter configuration of a certain algorithm (e.g.,

tolerance in Newton's method). The *ModelType* class represents the type of the optimization model. Sub-types of *ModelType* can include *LinearProgramming*, *IntegerProgramming*, *NonlinearProgramming*, etc.

To formally represent mathematical expressions in the Optimization Metamodel, MathML has been used to capture the mathematical expressions of *Constraint*s and *ObjectiveFunction*s. Here, the Content ML, which represents the underlying mathematical structure of an expression, is chosen to represent the mathematical expressions in the MILP model.

5.1.2.2 Representation of an MILP Model Using the Optimization Metamodel

In this paper, an FJSP problem is used to illustrate the utilization of the Optimization Metamodel. The problem description and mathematical modeling are selected from Özgüven et al. (2010). An FJSP consists of a set of n independent jobs $J = \{j_i\}_{i=1}^{n}$, each having its own processing order through a set of m machines $M = \{m_k\}_{k=1}^{m}$. A number of ℓ_i ordered operations $(O_{i1}, \dots, O_{i\ell_i})$ need to be performed to complete job i. Operation j of job $i(O_{ij})$ can be processed by any machine in a given set $M_j \subseteq M$ for a given processing time t_{ijk}. The FJSP is a routing as well as a sequencing problem: assigning each operation O_{ij} to a machine selected from the set M_j and ordering operations on the machines so that C_{max} (i.e. makespan) is minimized.

The following notation is used for the MILP model.

Indices and sets

i the index of jobs $(i, i' \in J)$

j the index of operations ($j, j' \in O$)

k the index of machines ($k \in M$)

J the set of jobs

M the set of machines

O the set of operations

O_i ordered set of operations of job i ($O_i \subseteq O$), where $O_{if_{(i)}}$ is the first and $O_{i\ell_{(i)}}$ is the last element of O_i

M_j the set of alternative machines on which operation j can be processed,($M_j \subseteq M$)

$M_j \cap M_{j'}$ the set of machines on which operations j and j' can be processed

Parameters

t_{ijk} the processing time of operation O_{ij} on machine k

L a large number

Decision variables

X_{ijk} 1, if machine k is selected for operation O_{ij}; 0, otherwise

S_{ijk} the starting time of operation O_{ij} on machine k

C_{ijk} the completion time of operation O_{ij} on machine k

$Y_{iji'j'k}$ 1, if operation O_{ij} precedes operation $O_{i'j'}$ on machine k; 0, otherwise

C_i the completion time of job i

C_{max} maximum completion time over all jobs (makespan)

The MILP model is defined as follows:

$Objective\ function$: Minimize C_{max}

$Constraints$:

$$\sum_{k \in M_j} X_{ijk} = 1 \quad \forall i \in J, \forall j \in O_i, \tag{1}$$

$$S_{ijk} + C_{ijk} \leq \left(X_{ijk}\right) \cdot L \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{2}$$

$$C_{ijk} \geq S_{ijk} + t_{ijk} - \left(1 - X_{ijk}\right) \cdot L \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j, \tag{3}$$

$$S_{ijk} \geq C_{i'j'k'} - \left(Y_{iji'j'k}\right) \cdot L \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'}, \tag{4}$$

$$S_{i'j'k'} \geq C_{ijk} - \left(1 - Y_{iji'j'k}\right) \cdot L \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'}, \tag{5}$$

$$\sum_{k \in M_j} S_{ijk} \geq \sum_{k \in M_j} C_{i,j-1,k} \quad \forall i \in J, \forall j \in O_i - \{O_{if_{(i)}}\}, \tag{6}$$

$$C_i \geq \sum_{k \in M_j} C_{i,O_{i\ell_{(i)}},k} \quad \forall i \in J, \tag{7}$$

$$C_{max} \geq C_i \quad \forall i \in J, \tag{8}$$

and

$$X_{ijk} \in \{0,1\} \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j,$$

$$S_{ijk} \geq 0 \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j,$$

$$C_{ijk} \geq 0 \quad \forall i \in J, \forall j \in O_i, \forall k \in M_j,$$

$$Y_{iji'j'k} \in \{0,1\} \quad \forall i < i', \forall j \in O_i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'},$$

$$C_i \geq 0 \quad \forall i \in J.$$

Constraint (1) makes sure that operation O_{ij} is assigned to only one machine. If operation O_{ij} is not assigned to machine k, constraint (2) sets its starting and completion times on machine k to zero. Otherwise, constraint (3) guarantees that the differences between the starting

and the completion times is at least equal to the processing time on machine k. Constraints (4) and (5) fulfill the requirement that operation O_{ij} and operation $O_{i'j'}$ cannot be carried out at the same time on any machine in the set $M_j \cap M_{j'}$. Constraint (6) captures the precedence relationships between the operations of a job, i.e. the operation O_{ij} cannot start before the operation $O_{i,j-1}$ has been completed. Constraint (7) determines the completion times (of the final operations) of the jobs. Constraint (8) determines the makespan.

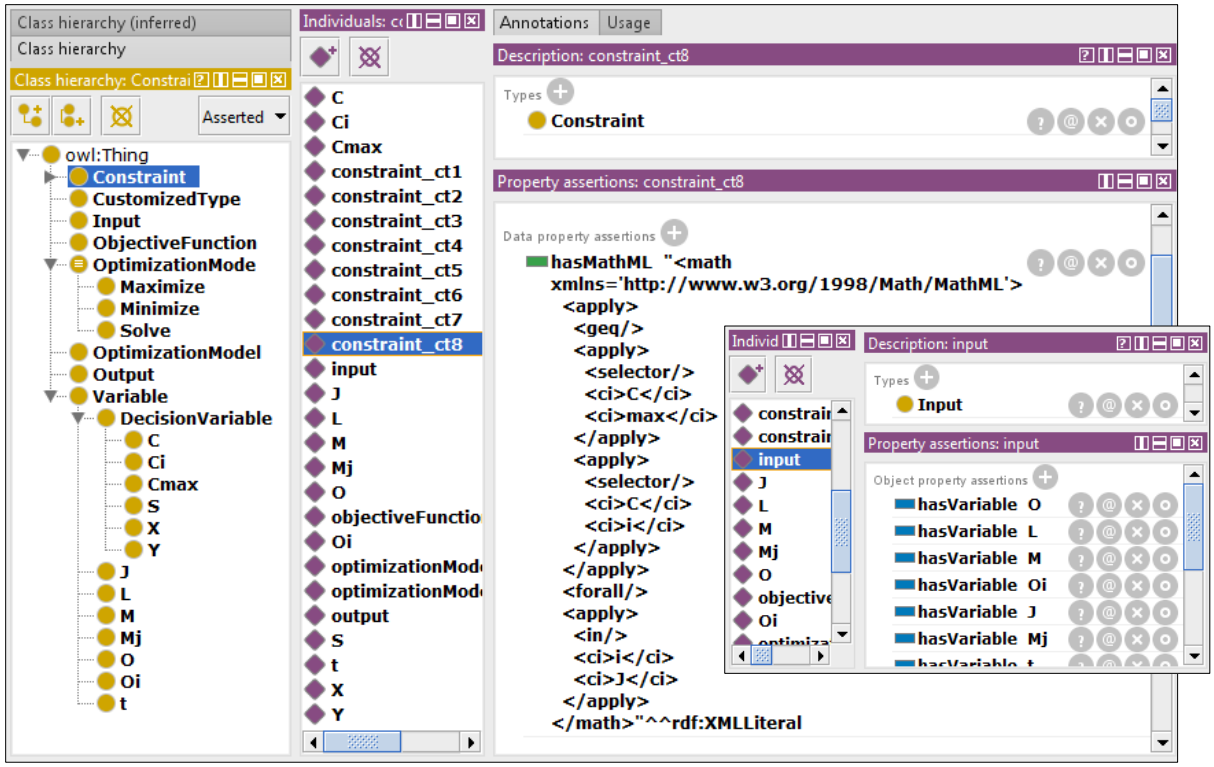


Figure 5.3 Representation of the MILP model using the Optimization Metamodel in protégé 5.2

To show an example of using the Optimization Metamodel to represent this MILP model, the OWL language has been used. To capture the MILP model, the Optimization Metamodel is first expanded to include the variables (e.g., X_{ijk}, S_{ijk}, and C_{ijk}, etc.), constraints (e.g., constraint (1)

to (8)), and objective function. Figure 5.3 shows a representation of the populated Optimization Metamodel for the MILP model in protégé 5.2. An example of MathML representation for constraint (8) is demonstrated in the figure. The relationships between a class and a member field (Figure 1) are modeled as "has~" object properties. An example of the *hasVariable* object property of the *input* instance is shown in the mini-window of Figure 5.3.

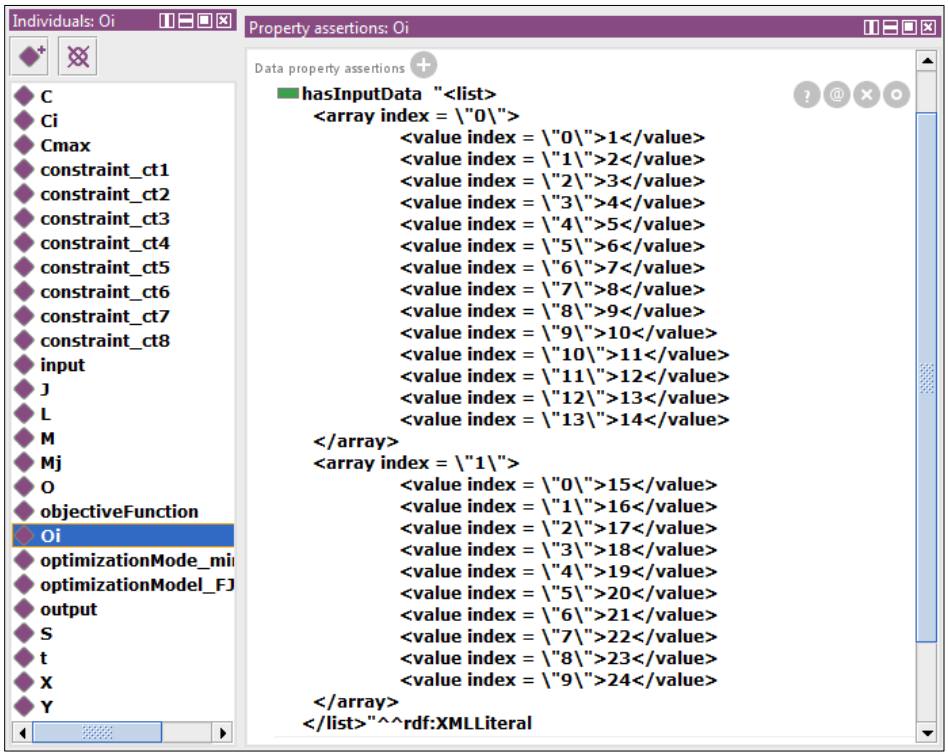


Figure 5.4 XML representation of input data for variable $\boldsymbol{O_i}$

The built-in datatypes of OWL have been used to represent the datatypes of the decision variables. For example, the X_{ijk} and $Y_{iji'j'k}$ variables are represented as *xsd:Boolean* because they only take values as 0 or 1. Decision variables S_{ijk}, C_{ijk}, and C_i are represented as *xsd:nonNegativeInteger*. Since OWL does not have primitive types for the multi-dimensional

array, XML is used to represent arrays for input and output data in this work. An example of the input data for variable O_i is shown in Figure 5.4.

5.1.3 Development of a Knowledge Enriched Optimization Model for Model Deployment

To demonstrate the representational capability of the proposed Optimization Metamodel, the Optimization Metamodel for a Constraint Programming (CP) model is developed. The CP model, which solves the FJSP problem discussed in section 5.1.2.2, is selected from an example model provided by IBM Cplex studio.

The notation of the model is described below.

Decision variables

Ops the array of operation intervals

$Modes$ the array of alternative operation intervals on each machine for all the operations

$Mchs$ the array of machine schedule sequences

Indices and sets

i the index of Ops ($i \in [1, count(Ops)]$)

j the index of $Modes$ ($j \in [1, count(Modes)]$)

k the index of $Mchs$ ($k \in [1, count(Mchs)]$)

Parameters

$opId(Ops_i)$ the id of an operation; starting from 1

$jobId(Ops_i)$ the job id of an operation; starting from 1

$pos(Ops_i)$ the position of an operation in a job; starting from 0

$opId(Mode_j)$ the operation id of an alternative operation; the alternative operations of an operation Ops_i can be identified by $opId(Mode_j)$

$mch(Mode_j)$ the machine id of an alternative operation

$pt(Mode_j)$ the processing time of an alternative operation

The CP model is defined as follows:

$Objective\ function$:

Minimize

$$max(\{end(Ops_i)\}) \quad i \in \{i | \forall j \in J \cap max(\{pos(Ops_i) | jobId(Ops_i) = j\})\}$$

$Constraints$:

$$endBeforeStart(Ops_i, Ops_{i'}) \quad \forall i' = i + 1, jobId(Ops_i) = jobId(Ops_{i'}), \qquad (1)$$

$$alternative(Ops_i, \{Modes_j | opId(Modes_j) = opId(Ops_i)\}), \qquad (2)$$

$$noOverLap(Mchs_k). \qquad (3)$$

Constraint (1) captures the precedence relationships between the operations. Constraint (2) represents the alternative operation intervals that an operation can select from. Constraint (3) makes sure that the operation intervals within a machine schedule do not overlap.

To demonstrate the utilization of the KECM, this section describes the development of a Knowledge Enriched Optimization Model. An Optimization Metamodel which represents a Constraint Programming model has been used as an example to show the enrichment. In the

following sections, the development and utilization of the information model and the rationales/rules used in the Knowledge Enriched Optimization Metamodel are illustrated. In this case study, the Optimization Metamodel, the information model, and the rationales/rules are implemented using OWL. The rationales are implemented using the SWRL in OWL.

5.1.3.1 Information Model

The information model used in this paper is selected from a previous work (Zhang et al. 2015). This information model was developed to facilitate sustainability evaluation in the manufacturing domain. A compact version of the information model, or the Sustainable Manufacturing Ontology (SMO), is shown in Figure 5.5. A brief explanation of some important concepts in the information model is narrated below:

- **Equipment**: *Equipment* can be a tool or a machine on the shop floor.
- **Shop**: A Shop represents a manufacturing facility in a factory. It has a set of Jobs that are to be finished. A Shop has a variety of Equipment that is used to carry out the Jobs.
- **Job**: A *Job* defines a task that needs to be carried out to produce a *Part*. Each *Job* is composed of a series of *Operation*s.
- **Operation**: An *Operation* represents the task to be performed to produce a feature of a *Part*. Each *Operation* contains a *ManufacturingProcess*. An *Operation* utilizes a certain type of *Equipment* to carry out its task.

  For more descriptions of the SMO, please refer to section 4.4.1.

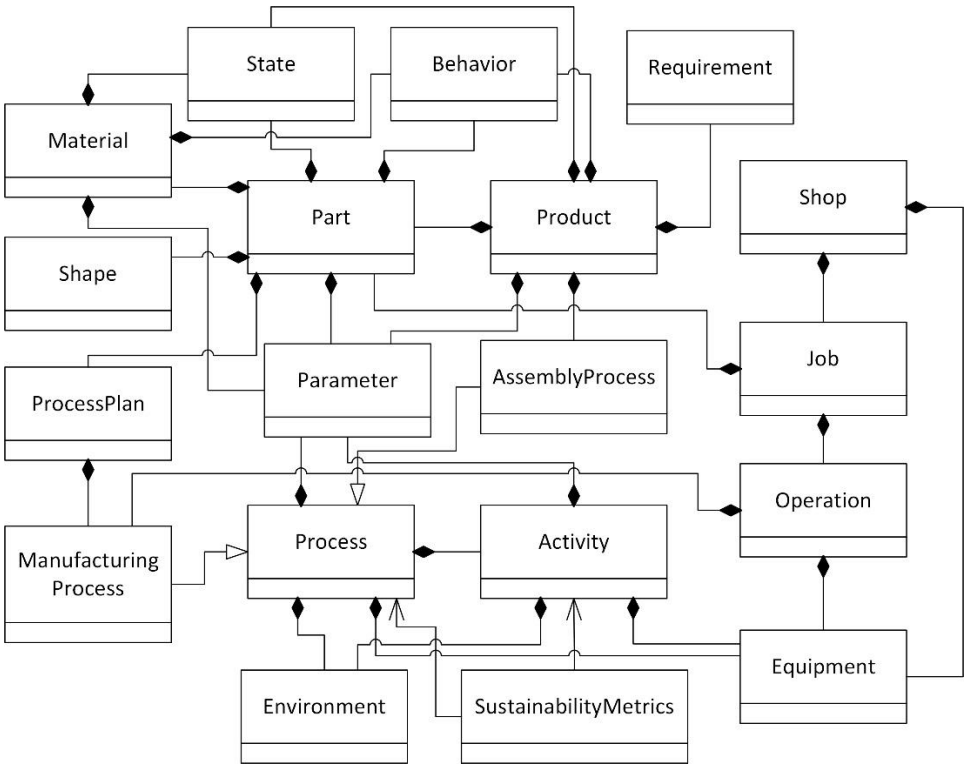


Figure 5.5 A UML representation of the extended Sustainable Manufacturing Ontology (SMO)

5.1.3.2 Optimization Metamodel

The Optimization Metamodel has been expanded to represent this CP model (Figure 5.6). OWL is also used to represent the Optimization Metamodel. In Figure 5.6, the black boxes represent *owl:class*es; the green boxes are datatypes; the pink arrows indicate the *hasSubClass* relationships; the red arrows indicate the 'has-a' object properties; the green arrows indicate the data properties. Other than capturing the constraints and variables in the CP model, three *CustomizedType*s have been defined. An *Interval* is an entity that has a start time, end time, and processing time. A *Sequence* is an entity that represents a schedule for a machine. It is composed of an ordered set of *Interval*s. An *AlternativeMachineSet* represents the alternative machine set of an operation. It is composed of a set of *Mode*s.

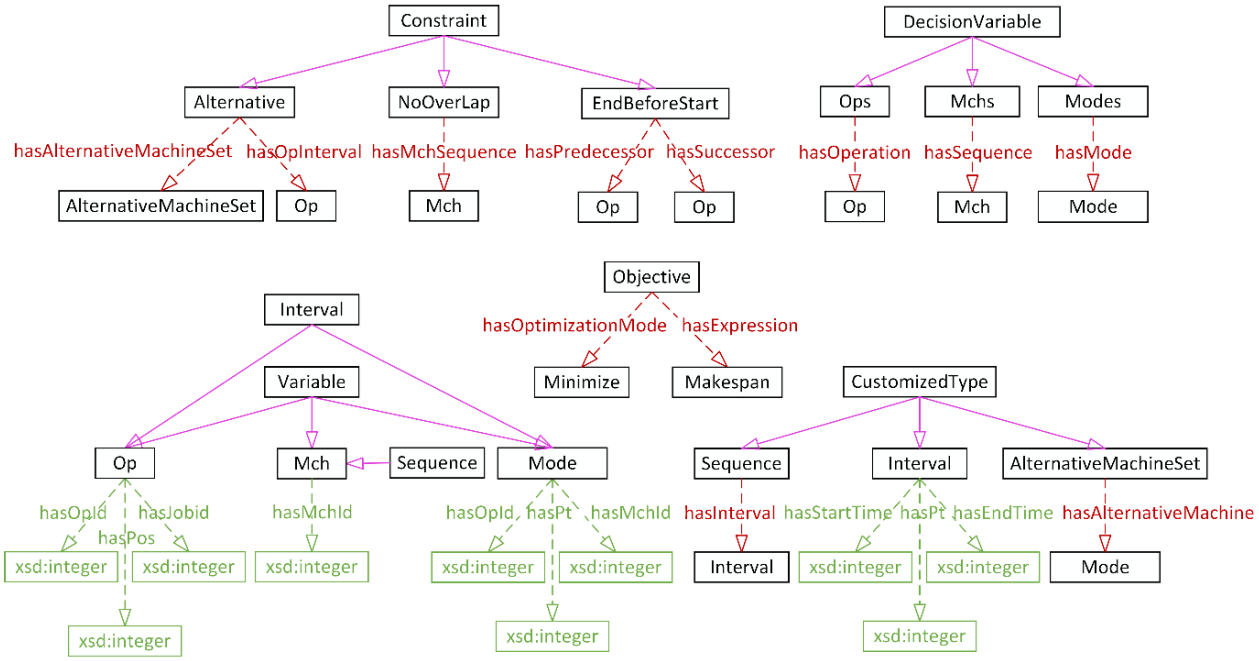


Figure 5.6 Expansion of the Optimization Metamodel with respect to the CP model

5.1.3.3 Rationales

This section presents examples to capture rationales/rules for the deployment and reuse of the CP model presented in the last section. The SWRL language in OWL has been used to represent the rationales/rules.

As discussed in section 5.1.1, to facilitate the deployment of an optimization model, this paper proposes to first load data from the underlying SM system to the information model. Then, the data stored with the information model are loaded to the Optimization Metamodel. The following rationales/rules have been developed for capturing the mappings between the information model and the Optimization Metamodel.

Load *jobId* for *Op* from Information Model

Job(?job), xsd:integer(?jobId), hasId(?job, ?jobId), Operation(?operation), hasOperation(?job, ?operation), xsd:integer(?opId), hasId(?operation, ?opId), Op(?op), hasOpId(?op, ?opId) -> hasJobId(?op, ?jobId)

The meaning of this rule is: If a *Job* has id *?jobId* and it has an *Operation* which has id *?opId*, the *Op* which also has id *?opId* should have job id *?jobId*.

Load *pos* for *Op* from Information Model

Operation(?operation), xsd:integer(?id), hasId(?operation, ?id), Op(?op), hasOpId(?op, ?id), xsd:integer(?position), hasPosition(?operation, ?position) -> hasPos(?op, ?position)

The meaning of this rule is: For an *Operation* that has id *?id* and has a position *?position*, the corresponding *Op* which has the same id *?id* should also have a position *?position*.

Load *mchId* for *Mode* from Information Model

Operation(?operation), AlternativeOperation(?altOperation), hasAlternativeOperation(?operation, ?altOperation), xsd:integer(?altOptId), hasId(?altOperation, ?altOptId), Process(?process), hasProcess(?altOperation, ?process), xsd:integer(?processType), hasProcessType(?process, ?processType), Mode(?mode), hasModeId(?mode, ?altOptId) -> hasMchId(?mode, ?processType)

The meaning of this rule is: If an *Operation* has an *AlternativeOperation* which has id *?altOptId*, and the *Process* of the *AlternativeOperation* has process type *?processType*, the

Mode (in the CP model) which has the same id as the *AlternativeOperation* should have *MchId ?processType*.

Load *pt* for *Mode* from Information Model

Operation(?operation), AlternativeOperation(?altOperation),
hasAlternativeOperation(?operation, ?altOperation), xsd:integer(?processingTime),
xsd:integer(?altOptId), hasId(?altOperation, ?altOptId),
hasProcessingTime(?altOperation, ?processingTime), Mode(?mode),
hasModeId(?mode, ?altOptId) -> hasPt(?mode, ?processingTime)

The meaning of this rule is: If an *Operation* has an *AlternativeOperation* which has id *?altOptId*, and the *Process* of the *AlternativeOperation* has processing time *?processingTime*, the *Mode* (in the CP model) which has the same id as the *AlternativeOperation* should have *pt ?processingTime*.

Rationales are the reasons or descriptions about why or how a model is developed are developed for the case study. In this case study, the rationales that formally define the semantics of the three constraints have been developed.

Constraint (1) endBeforeStart

This constraint defines the precedence relationships between the adjacent operations in a job. Two rules have been individually developed to capture the predecessor and/or successor of an operation since the first/last operation of each job only has a successor/predecessor. This constraint is defined such that an operation *Op* has a constraint *endBeforeStart*, and the constraint

endBeforeStart captures *Op*'s predecessor and/or successor. The *swrlb:add* and the *swrlb:equa*l relationships are the built-in relationships in the SWRL language. *swrlb:add* is satisfied if and only if the first argument is equal to the arithmetic sum of the second argument through the last argument. *swrlb:equal* is satisfied if and only if the first argument and the second argument are the same.

Op(?op), Op(?op1), Ops(dvar_ops), endBeforeStart(?endBeforeStart), xsd:integer(?jobId), xsd:integer(?jobId1), xsd:integer(?pos), xsd:integer(?pos1), hasConstraint(?op, ?endBeforeStart), hasOp(?ops, ?op), hasOp(?ops, ?op1), hasJobId(?op, ?jobId), hasJobId(?op1, ?jobId1), hasPos(?op, ?pos), hasPos(?op1, ?pos1), swrlb:add(?pos, 1, ?pos1), swrlb:equal(?jobId, ?jobId1) -> hasPredecessor(?endBeforeStart, ?op1)

The meaning of this rule is: There are two *Op*s in *dvar_ops: ?op* and *?op1*. They have the same *jobId*. The position of *?op* is greater than that of *?op1* by 1. So, the *endBeforeStart* constraint of *?op* should have a predecessor *?op1*.

Op(?op), Op(?op1), Ops(dvar_ops), endBeforeStart(?endBeforeStart), xsd:integer(?jobId), xsd:integer(?jobId1),xsd:integer(?pos),xsd:integer(?pos1),hasConstraint(?op, ?endBeforeStart) , hasOp(?ops, ?op), hasOp(?ops, ?op1), hasJobId(?op, ?jobId), hasJobId(?op1, ?jobId1), hasPos(?op, ?pos), hasPos(?op1, ?pos1), swrlb:add(?pos1, 1, ?pos), swrlb:equal(?jobId, ?jobId1) -> hasSuccessor(?endBeforeStart, ?op1)

This rule is similar to the previous one. It adds the successors for any *Op*.

Constraint (2) alternative

This constraint defines the alternative machines for a certain operation. The first rule defines a *CustomizedType – AlternativeMachineSet*. One *AlternativeMachineSet* is created for each *Op*. The *Mode*s, which have the same *opId* as an *Op* does, are added to the corresponding *AlternativeMachineSet*. The second rule captures the relationship between an *Op* and its *AlternativeMachineSet* in an *alternative* constraint.

```
AlternativeMachineSet(?altMachineSet), Mode(?mode), Modes(dvar_modes), Op(?op),
xsd:integer(?opId), hasMode(dvar_modes, ?mode), hasOpId(?mode, ?opId),
hasOpId(?op, ?opId), hasOpId(?altMachineSet, ?opId) ->
hasAlternativeMachine(?altMachineSet, ?mode)
```

The meaning of the first rule is: For an *AlternativeMachineSet* that has the same *opId* as a *Mode*, the *AlternativeMachineSet* should have *Mode* as one of its *AlternativeMachine*.

```
AlternativeMachineSet(?altMachineSet), Op(?op), hasConstraint(?op, ?alternative),
alternative(?alternative), xsd:integer(?opId), hasOpId(?altMachineSet, ?opId),
hasOpId(?op, ?opId) -> hasAlternativeMachineSet(?alternative, ?altMachineSet),
hasOpInterval(?alternative, ?op)
```

The meaning of the second rule is: For an *AlternativeMachineSet* that has the same *opId* as an *Op*, the *alternative* constraint of the *Op* should be related to the *AlternativeMachineSet*.

Constraint (3) noOverlap

This constraint defines that within a schedule (i.e. a *Sequence*), no *Interval*s can be overlaps. The first rule defines that each *Mch* is composed of a set of alternative operations – *Mode*s, which have the same *mchId* as *Mch* does. The second rule describes that the *noOverlap* constraint is applied to every individual *Mch* in the whole schedule.

Mch(?mch), Mchs(dvar_mchs), Mode(?mode), Modes(dvar_modes), xsd:integer(?mchId1), xsd:integer(?mchId2), hasMch(dvar_mchs, ?mch), hasMode(dvar_modes, ?mode), hasMchId(?mch, ?mchId1), hasMchId(?mode, ?mchId2), swrlb:equal(?mchId1, ?mchId2) -> hasInterval(?mch, ?mode)

The meaning of this rule is: For any *Mch* (in *Mchs*) has the same *mchId* with a *Mode* in *dvar_modes*, the *Mch* should have the *Mode* as its *Interval*.

Mch(?mch), Mchs(dvar_mchs), noOverlap(?noOverlap), hasMch(dvar_mchs, ?mch) -> hasMchSequence(?noOverlap, ?mch)

The meaning of this rule is: The constraint *noOverlap* should have all *Mch*s in *dvar_mchs* in its *MchSequence*.

5.1.4 Utilization of the Knowledge Enriched Optimization Model for Model Deployment

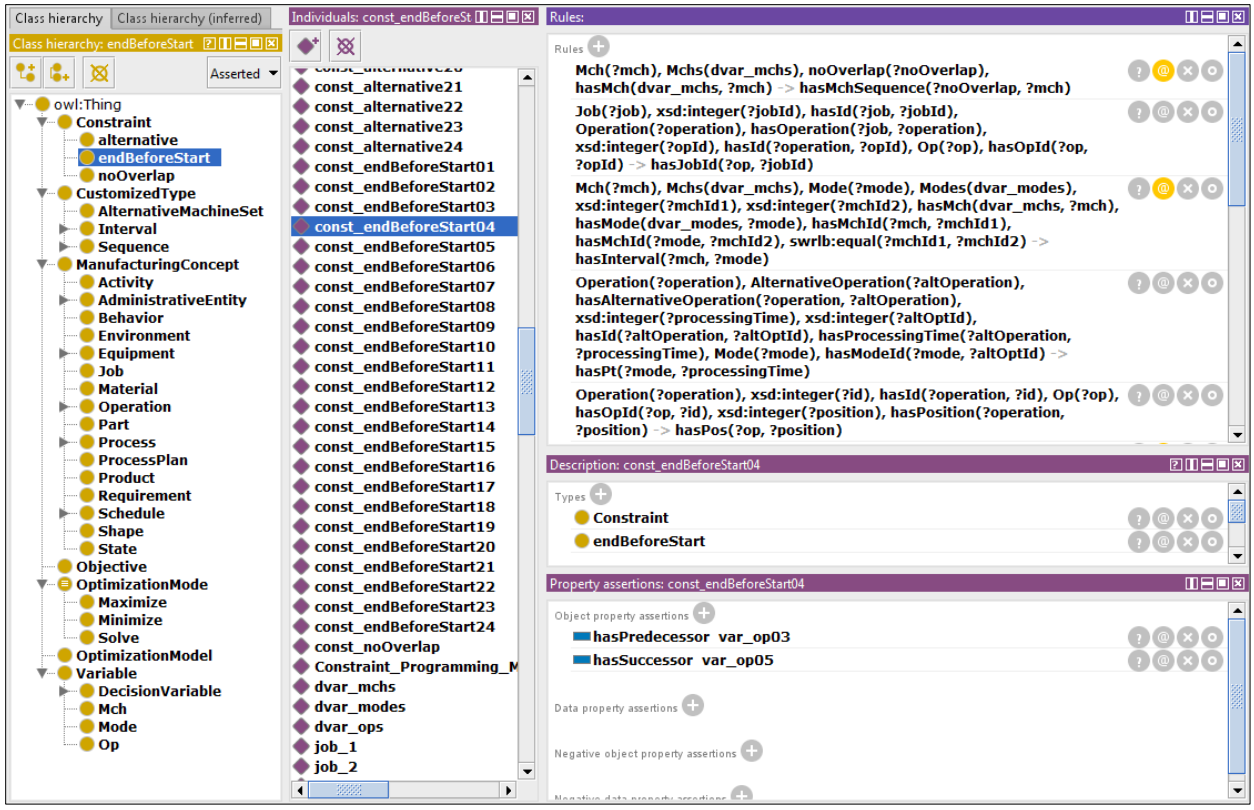


Figure 5.7 Screenshot of the implemented Optimization Metamodel in protégé 5.2

Figure 5.7 shows a representation of the developed Enriched Optimization Metamodel for the CP model. The information model, Optimization Metamodel, and rationales are all represented in protégé 5.2. This section discusses the utilization of the Enriched Optimization Metamodel from two perspectives: interoperability enabled by the Optimization Metamodel and deploying the Optimization Metamodel in a manufacturing system.

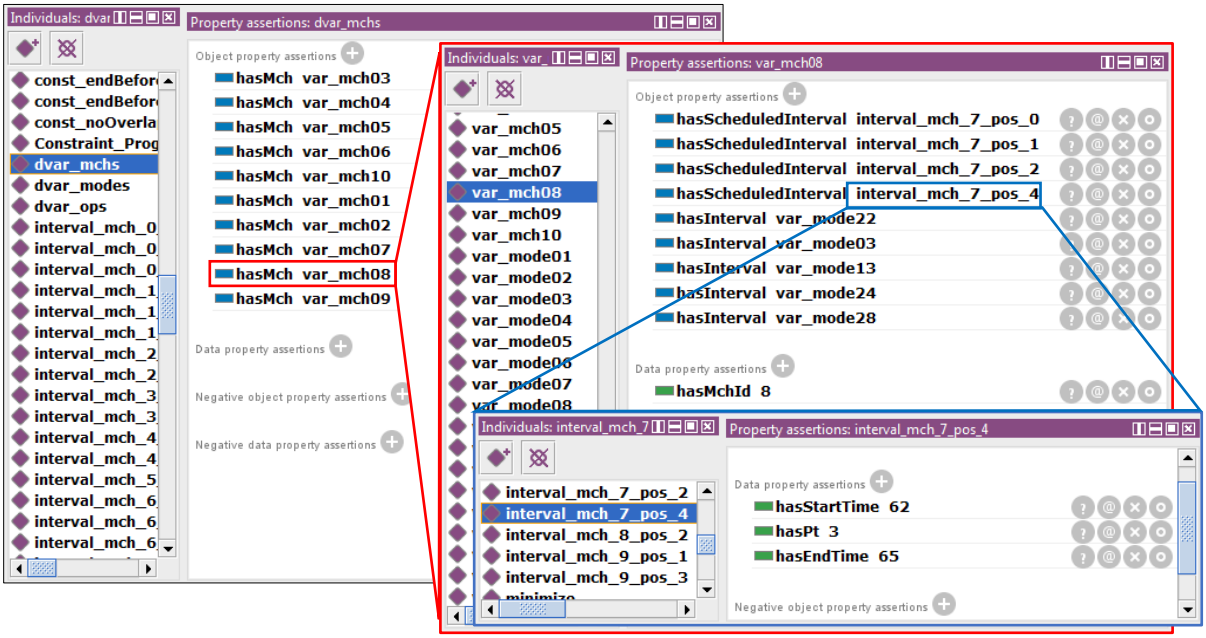


Figure 5.8 Representation of the optimization result (schedule) in protégé 5.2

5.1.4.1. Interoperability Enabled by the Optimization Metamodel

In this case study, the interoperability of the CP model contained in the Optimization Metamodel has been tested. After the instantiated Optimization Metamodel has been developed, the metamodel is executed by two CP solvers: IBM Cplex CP solver and Google OR-Tools. The metamodel is consumed by the Java APIs (Application Programming Interface) of the two tools through a developed metamodel parser using OWLAPI (Horridge and Bechhofer, 2011). After the metamodel has been executed, the optimization result – the schedule – is loaded back to the metamodel (Figure 5.8) through the metamodel parser again. Both optimization results obtained from the two tools appear to be the same and correct. In this test, the Optimization Metamodel is proved to be capable of representing optimization models in a text-based format and is capable of supporting the interoperability of the optimization models among different optimization tools.

5.1.4.2. Using the KECM to Support Model Deployment

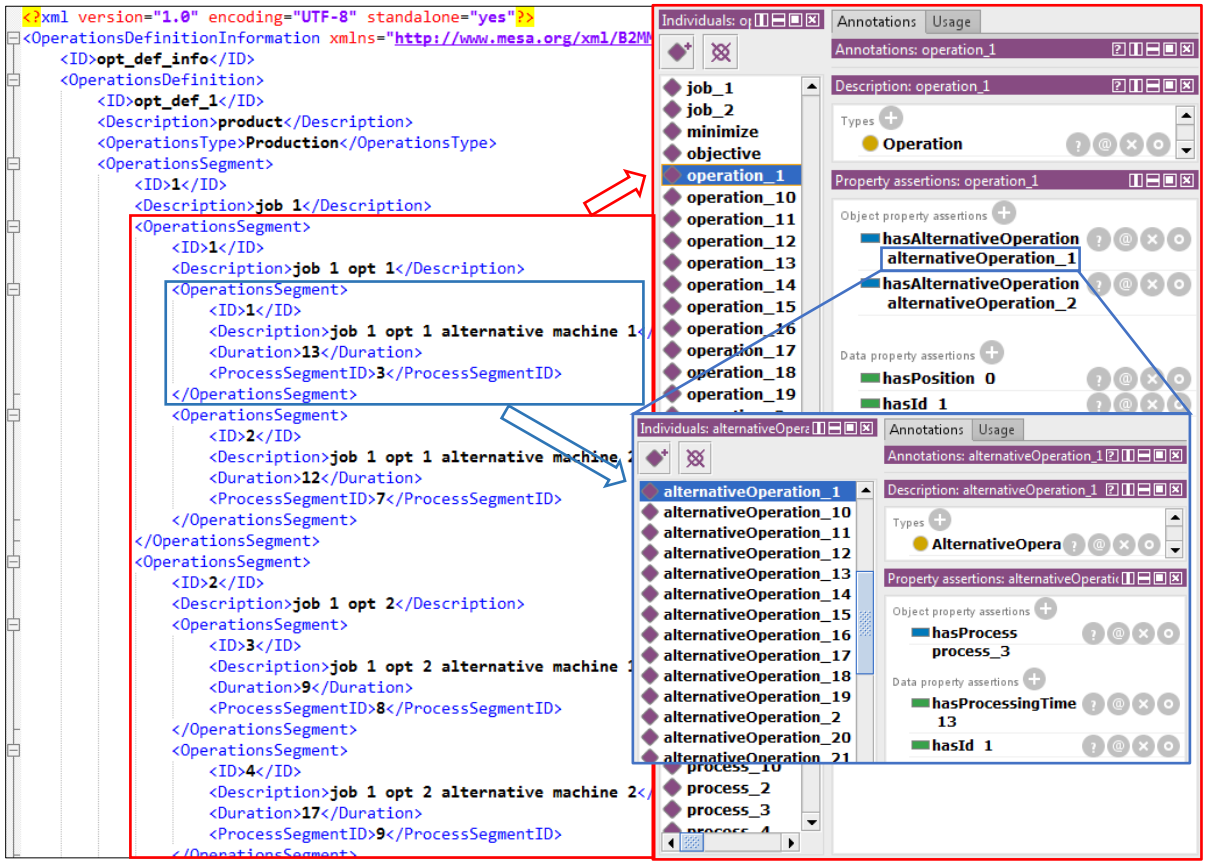


Figure 5.9 Input data in B2MML and in the Knowledge Enriched Optimization Model

The utilization of the Enriched Optimization Metamodel to support model deployment is demonstrated by the using the first set of rationales. A scenario of deploying the model in an ANSI/ISA-95-based scheduling system is assumed. In this scenario, data exchange between the underlying information system and the optimization solution is achieved using B2MML. The input data of the optimization model is imported from a B2MML-based XML file. Through a defined mapping file between the B2MML (i.e. the data model) and the SMO (i.e. the information model), the input data are first loaded to the SMO. Then, by turning on the reasoning

engine with the set of mapping rules (section 5.1.3.3), the data in the SMO are transferred onto the Optimization Metamodel. After consuming the CP model with a developed parser in IBM Cplex using its Java API, the scheduling result can be loaded back to the KECM again through the parser. An example of the input data in B2MML and in the Knowledge Enriched Optimization Metamodel is shown in Figure 5.9.

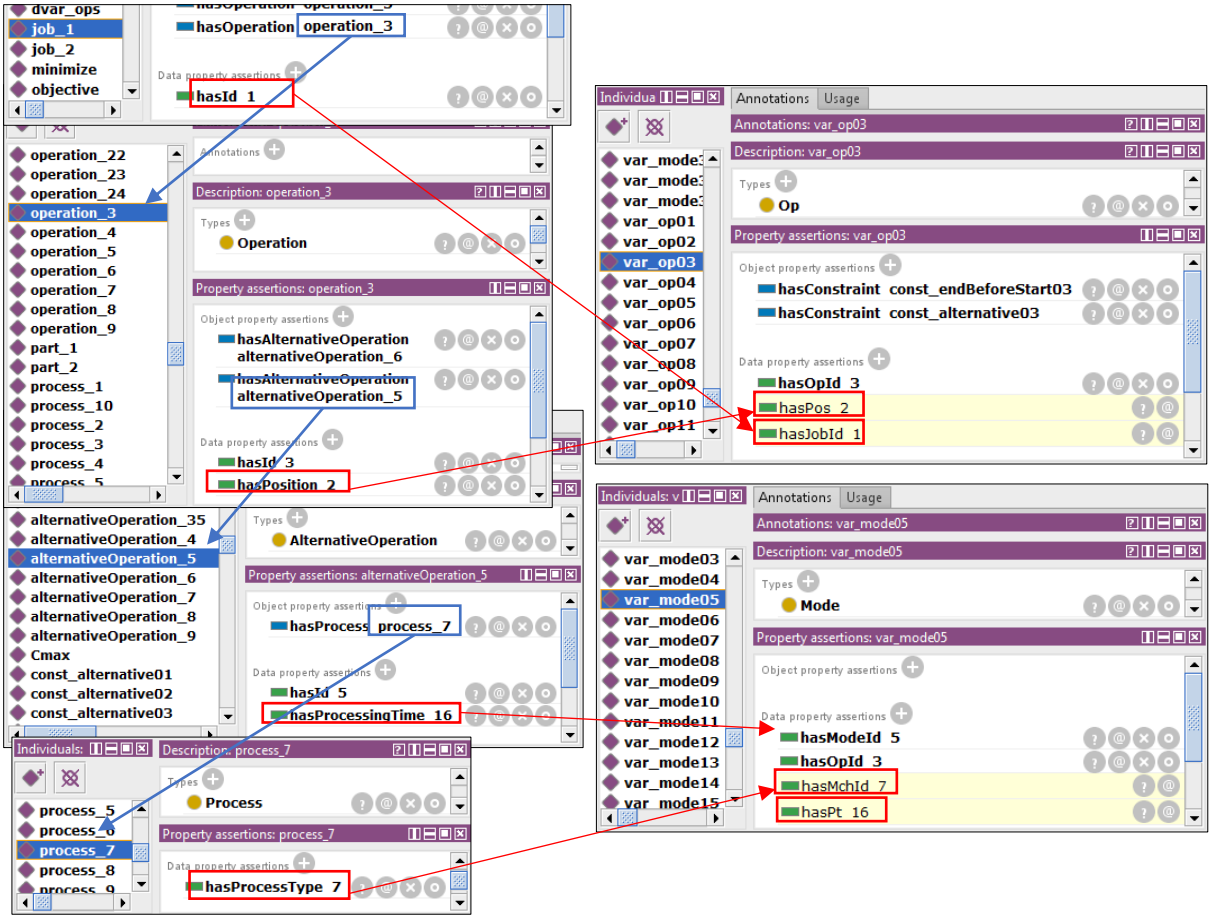


Figure 5.10 Loading the data from the SMO to the Optimization Metamodel

Figure 5.10 demonstrates the utilization of the rules to load data from the SMO to the Optimization Metamodel. The entities highlighted in light yellow are inferred using the pellet

reasoner in protégé 5.2. The red boxes and arrows indicate the related data in the SMO and the Optimization Metamodel.

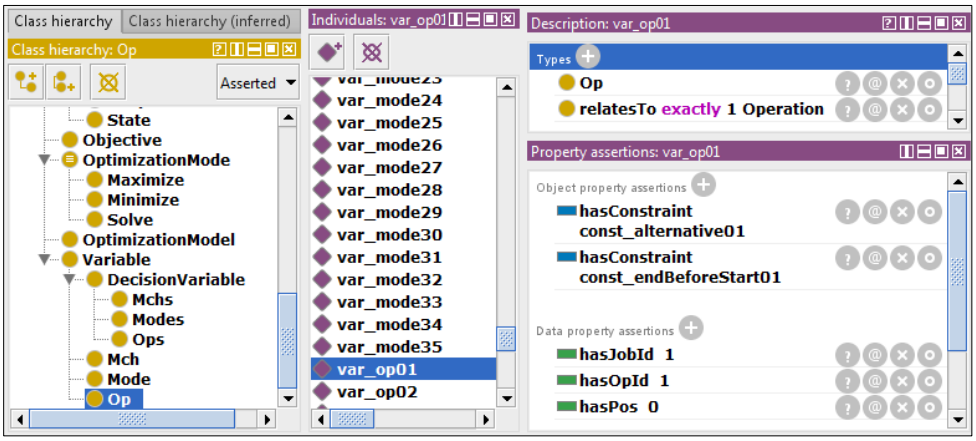


Figure 5.11 Representing domain meaning of optimization model's entities

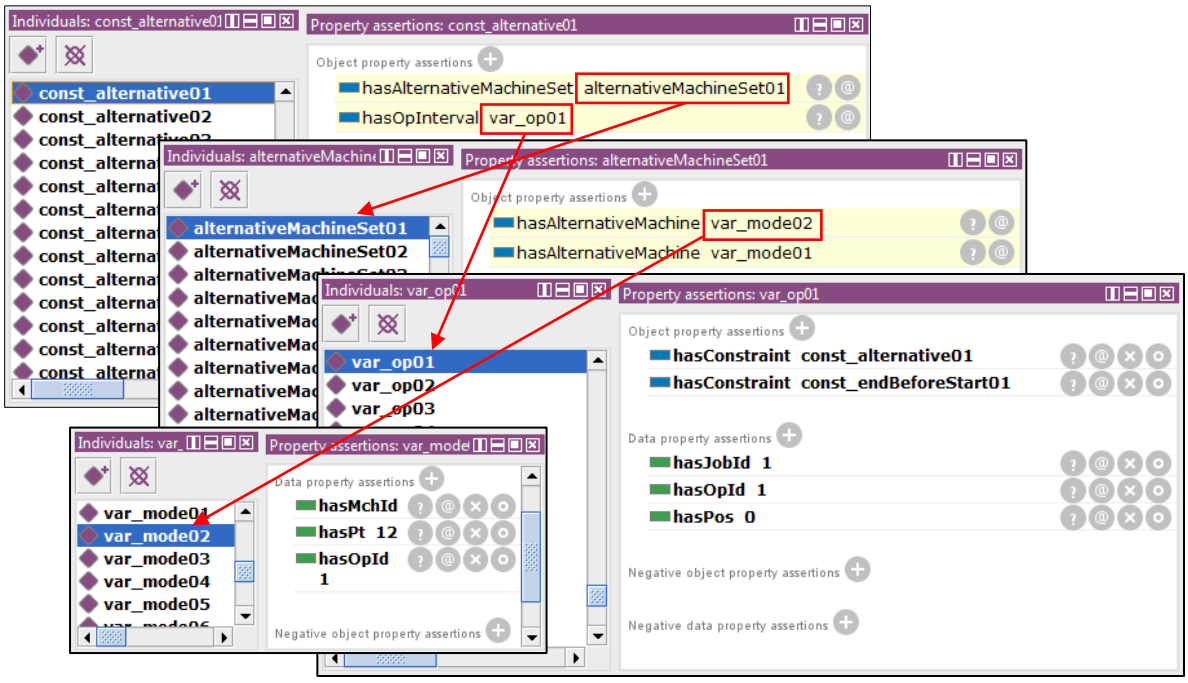


Figure 5.12 Generating constraint instances with the rationales

The knowledge, which captures the domain meanings and model explanations, implemented in this case study contains: expressing the domain meanings of variables using an information

model and generally capturing the semantics of the constraints in the CP model. Using the information model to express the domain meanings of variables is achieved by semantically connecting the variables defined in the Optimization Metamodel and the domain concept captured by the information model. An example of expressing the domain meanings of an *Op* variable using the *Operation* concept in the SMO is shown in Figure 5.11.

The generally defined semantics of the constraints in the CP model is described in section 5.1.3.3. Whenever the input data is loaded to the KECM, these sets of rationales can be executed to construct the constraint instances. The metamodel with the generated constraint instances can then be parsed and executed in optimization tools. Figure 5.12 shows an example of the constraint instances generated by reasoning the rationales.

5.2 A Methodology to Support the Combination of Computational Models

Normally, each computational model is developed to address a specific set of industrial issues, and it can only apply to a small portion of a complex Smart Manufacturing system. To allow the SM systems to solve more complex problems, individual computational models that were developed for different domain applications must be properly combined: (1) to simplify the original complex problem, individual computational models that solve different small problems can be combined to collaboratively solve the bigger problem of the systems; (2) to enhance computing performance, predictive models may be combined to reduce the prediction variance and bias; (3) if there are no dependencies between the individual computational models or the size of the data set is too large for one model to process, models can be combined to support

parallel computations. It is important to address the problem of model combination for the KECM. This is because a model combination represents the overall goal of a domain application; the sub-models can only represent the sub-goals to achieve the overall goal.

Thus, this chapter presents an approach to uniformly represent model combinations that are compatible with the KECM. To validate the proposed approach, a case study that combines an Agent-based model and a Decision Tree model has been developed for the utilization of the model combination representation in a real-time scheduling scenario.

5.2.1 Development of A Uniform Model for Model Combinations

Before uniformly representing model combinations, a general structure for computational models is defined (Figure 5.13). A model can be an individual computational model or a combined model that combines several individual models. A model should have its input(s) and output(s). Computational models can be normally combined in three methods: sequential models, parallel models, and composed models (Figure 5.14).

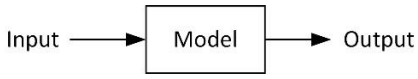


Figure 5.13 General structure for models

Sequential models are models that are combined sequentially: the outputs of one model are the inputs of another. The inputs of the combined model are the inputs of the first model, and the outputs of the combined model are the outputs of the last model.

Parallel models are parallelly combined. Depending on the application, the input data of the

combined model may be divided, and the divided data are consumed by the sub-models; the input data of the combined model can also be the same for all sub-models. The outputs from sub-models are normally combined according to the application. For example, if the outputs from the sub-models are all real numbers, the methods to combine the outputs can be weighted average, maximum, minimum, and summation, etc.

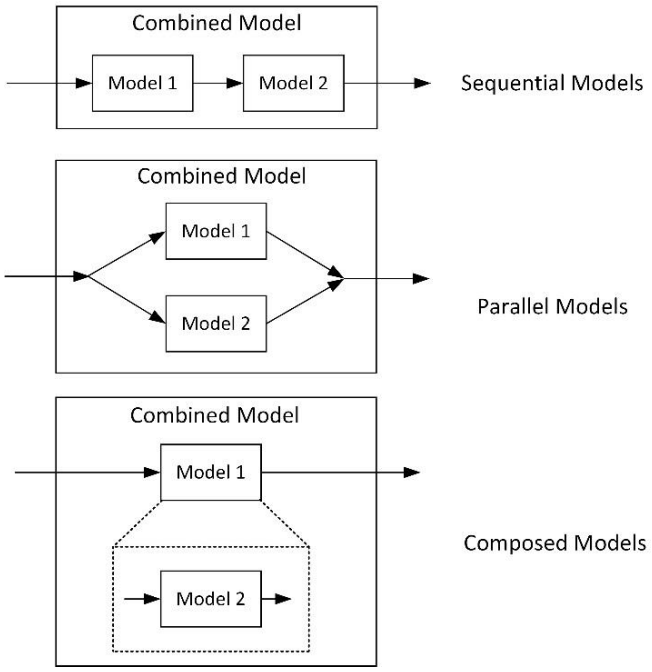


Figure 5.14 Methods for model combination

Composed models are models combined through composition. The functionality of one model is included in another model. The model being composed receives inputs from the external model; it outputs results to the external model.

More complex model combinations can be combined using these three basic model combination methods. Figure 5.15 demonstrates an example of the composition of combined models. In this figure, combined model 1 combines models (i.e. individual models or combined

models) by means of composed models; combined model 2 combines models through sequential models; combined model 3 combines models through parallel models.

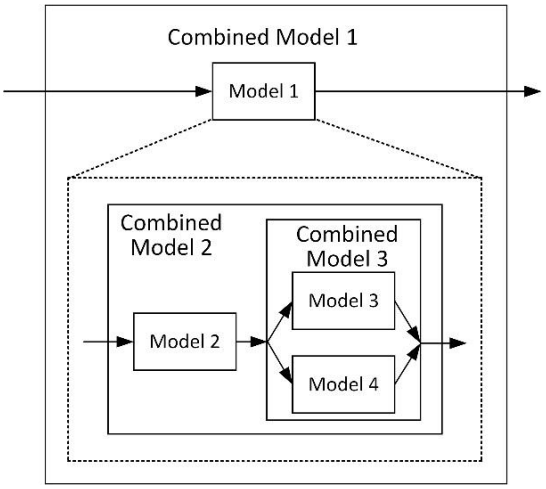


Figure 5.15 An example of the composition of combined models

A model composition representation has been developed to formally represent model compositions (Figure 5.16). Since the representation technology selected in this dissertation is OWL, this figure demonstrates representation that is compliant with OWL. In this figure, the purple arrows indicate the *hasSubClass* relationships; the red dashed arrows are *has-a* relationships.

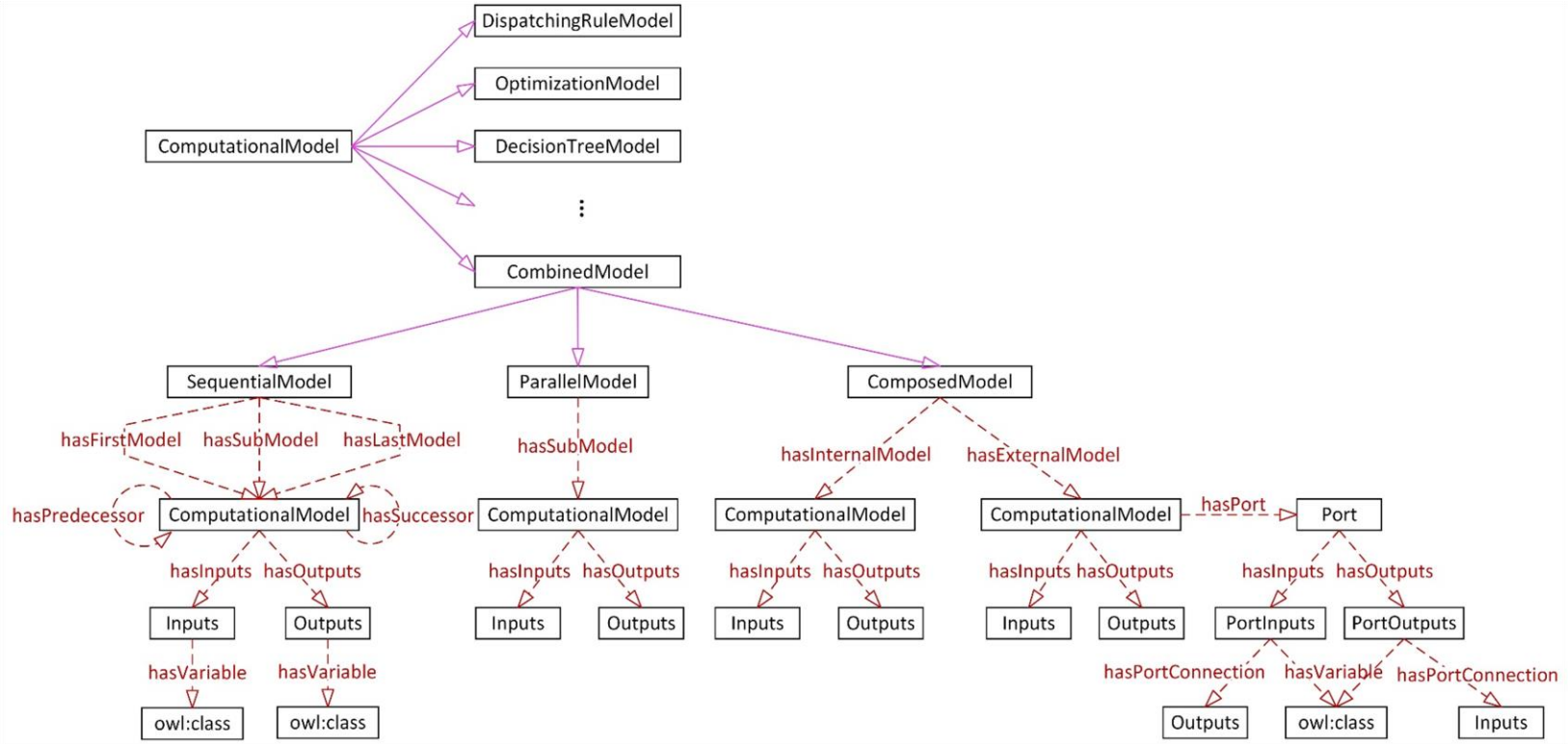


Figure 5.16 Representation of the general combination mechanisms

Other than the individual types of computational models like *OptimizationModel*, *DecisionTreeModel*, and *DispatchingRuleModel*, etc., *CombinedModel* is also considered as an inherited type of *ComputationalModel*. All *ComputationalModel*s should have *Inputs* and *Outputs* to represent the collections of input and output variables. Each variable which has a domain meaning and is contained in *Inputs* and *Outputs* is represented as an *owl:class*. The inputs-variable relationships are captured in the *hasVariable* relationships.

A *CombinedModel* has three sub-types: *SequentialModel*, *ParallelModel*, and *ComposedModel*. The *SequentialModel* has a sequence of *ComputationalModel*s. To represent the model sequence, each *ComputationalModel* can have *hasPredecessor* and *hasSuccessor* relationships to indicate its neighbor models. To specify the first and last model in a *SequentialModel*, the *hasFirstModel* and *hasLastModel* relationships can be used. For a model that is neither at the first nor the last position of the model sequence, its relationship within *SequentialModel* is represented using the *hasSubModel* relationships. The *ParallelModel* has a set of *ComputationlModel*s that are parallelly combined. Their relationships are represented using the *hasSubModel* relationships. The *ComposedModel* uses *hasExternalModel* and *hasInternalModel* relationships to indicate the external model and the internal model. For each external model, it has a *Port* to define the inputs/outputs between it and an internal model. Each *Port* has *PortInputs* and *PortOutputs* entities to denote the expected inputs from an internal model and the outputs to an internal model. Like *Inputs* and *Outputs*, the *PortInputs* and *PortOutputs* are entities to represent the collection of input and output variables. Each variable is also represented as an *owl:class* which has a domain meaning. To explicitly represent the

connections between the inputs and outputs of the composed models, relationship *hasPortConnection* can be used to connect *PortInputs* (or *PortOutputs*) of an external model to *Outputs* (or *Inputs*) of an internal model.

5.2.2 Case Study Scenario

To validate the developed model combination representation introduced in the last section, a case study has been developed to utilize a composed Agent-based model and a Decision Tree model in a real-time scheduling scenario. A real-time flexible job-shop scheduling scenario has been created based on an automated assembly line setup (Figure 5.17) selected from the literature (Trentesaux et al., 2013). In Figure 5.17, other than M5 and M6, all the workstations can carry out more than one type of job. Shuttles can travel between the workstations on a track following the arrow directions. The products produced by this assembly line are words formed by different parts (Figure 5.18). The parts are letters that are assembled using different shapes of components.

The scheduling scenario proposed in Trentesaux et al. (2013) has been extended in this case study. The parts made in the literature are only “A”, “B”, “E”, “I”, “L”, “P”, and “T”. In this case study, the parts have been expanded to all 26 English letters. Instead of the MILP problem presented in this literature, this case study adopts an Agent-based scheduling approach. This is because the Agent-based scheduling approach can rapidly respond to orders released in real-time although it cannot guarantee optimal solutions. For information about the production sequence of the 26 letters, please refer to the APPENDIX – A. For more information about the products and

assembly line configuration in this case study, please refer to Trentesaux et al. (2013).

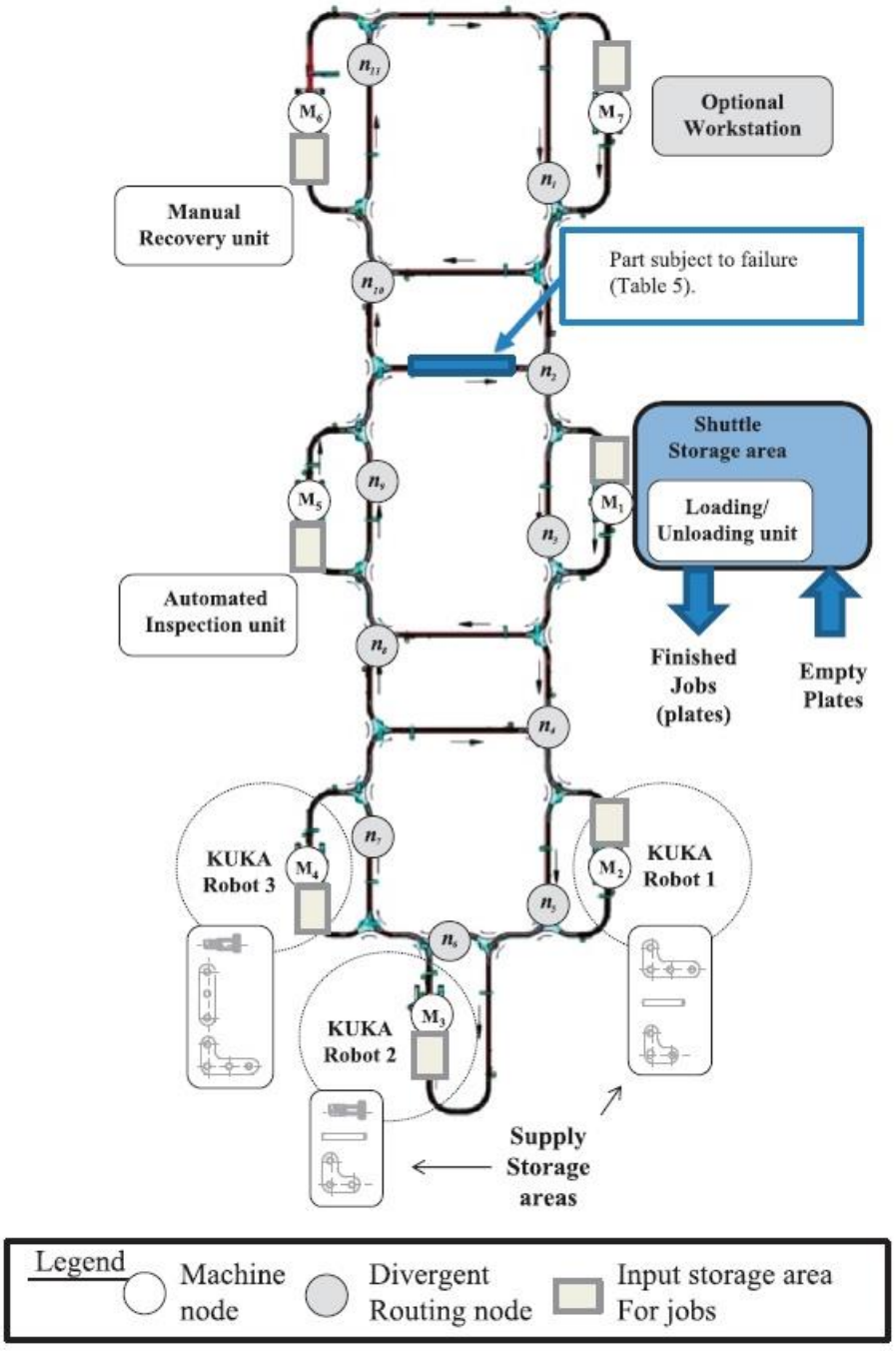


Figure 5.17 Shop floor layout of the real-time scheduling scenario (Trentesaux et al., 2013)

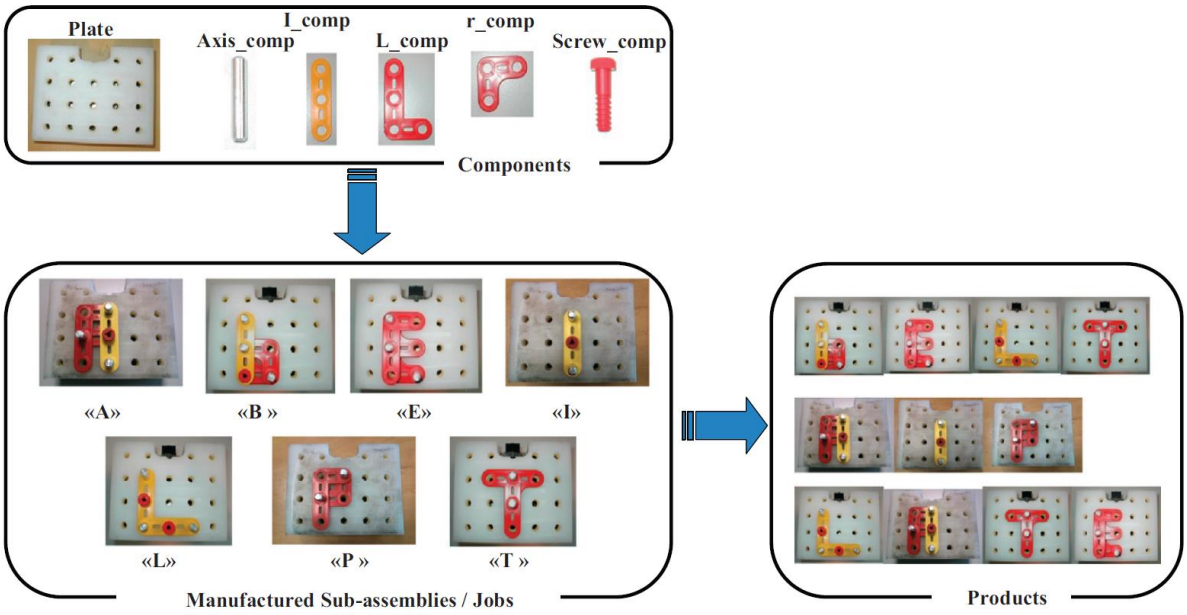


Figure 5.18 Components, jobs, and products produced on the production line (Trentesaux et al., 2013)

5.2.2.1 Development of A Composed Agent-based and Decision Tree System for Flexible Job Shop Scheduling

To achieve real-time scheduling for the assembly line discussed in the last section, an Agent-based system was first developed using an open source tool – JADE (Java Agent Development Framework) (Bellifemine et al., 2005). Four types of agents had been developed: shop floor agent, supervisor agent, product agent, and machine agent. The shop floor agent is responsible to monitor the status of the jobs and machines on the shop floor and to dispatch shuttles to workstations. The supervisor agent, product agents, and machine agents form a group to carry out scheduling and routing decisions. Each product agent represents a job (i.e. a letter) that needs to be assembled on the assembly line. Each machine agent represents a workstation on

the assembly line. The product agents and the machine agents are virtual entities that can communicate with each other and make decisions. The supervisor agent manages the activities of product agents and machine agents. It is responsible for instructing the shop floor agent to release and dispatch job shuttles to workstations.

Figure 5.19 demonstrates the basic system behavior. Once in every second, the shop floor agent sends a message (i.e. a red arrow in the diagram) to the supervisor agent, each product agent, and each machine agent to report the status of the shop floor. The *SUBSCRIBE* on each message is the communicative act which indicates the purpose of the message.

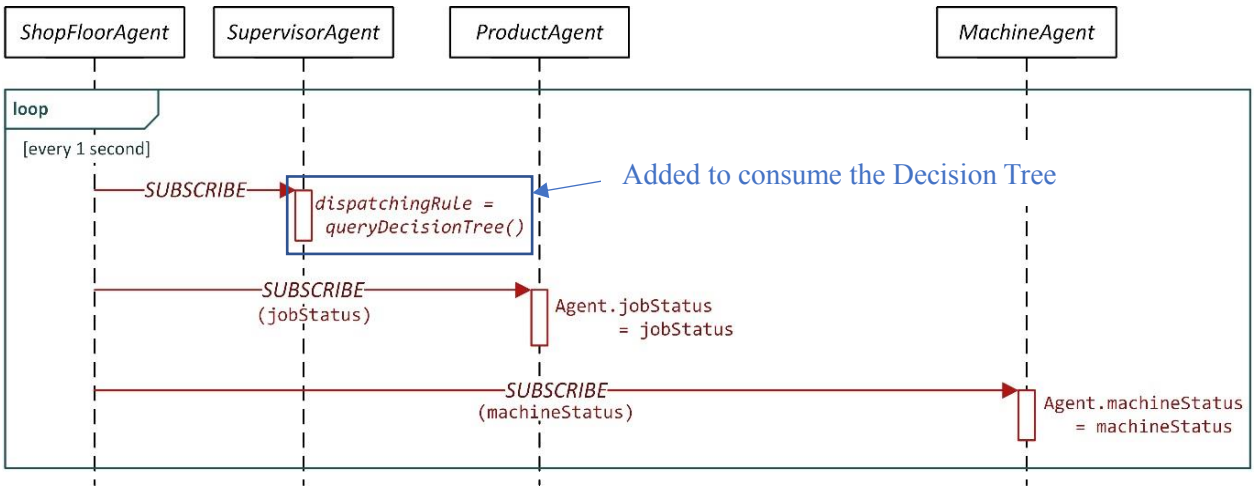


Figure 5.19 Sequence diagram to represent system behavior

Figure 5.20 shows the system behavior whenever a product order is released. Whenever the supervisor agent receives a *REQUEST* message which contains the requested product information (i.e. a list of letters that needs to make) from the outside of the Agent-based system, the supervisor agent will instruct the agent platform to create a set of product agents (i.e. each agent for a letter). Then, the supervisor agent informs the shop floor agent to release the raw

materials (i.e. the plate in Figure 5.18) for the parts. After a product agent has been created, it can receive the *CFP* (Call For Proposal) messages from the active machine agents. If the product agent is waiting and the machine(s) is capable of performing the activity requested by the part (e.g., assembling an L_comp onto the plate), it will calculate the priority value based on the dispatching rule selected, and send the priority value to the machine agent(s), which have just sent the *CFP* message, in a *PROPOSE* message. Otherwise, the product agent will send a *REFUSE* message to the machine agent. After each active machine agent receives all the *REFUSE* and *PROPOSE* messages, it evaluates all the proposals which contain the priority values sent from each product agent. After the machine agent has selected the best proposal with the lowest priority value, it replies to the product agent which has the best proposal with an *ACCEPT_PROPOSAL* message and sends other product agents *REJECT_PROPOSAL* messages. Then, each product agent which receives at least one *ACCEPT_PROPOSAL* message evaluates all the machines (which just accepted its proposal) and finds the nearest machine. Finally, the product agent replies to the nearest machine agent with an *INFORM* message and the others with *FAILURE* messages. At this point, the decision process for a product agent is completed. The whole process is developed based on the Contract Net Protocol defined by the FIPA standard (FIPA, 2002). After the decision, the product agent sends the machine allocation information to the supervisor agent, and then the supervisor agent instructs the shop floor agent to dispatch the job shuttle to go to the assigned machine.

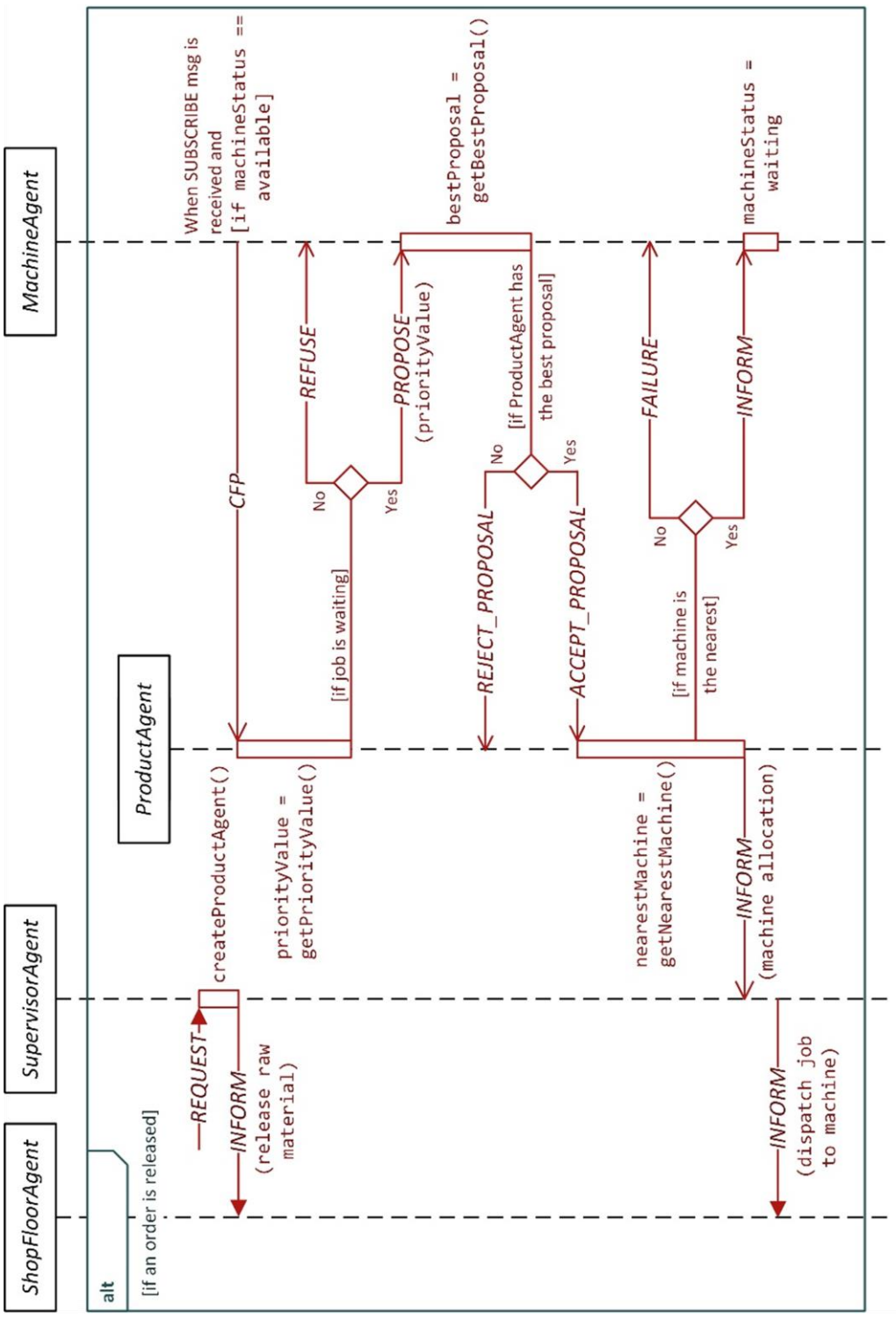


Figure 5.20 Sequence diagram to represent the system behavior when an order is released

This agent-based system was first developed to use a single dispatching rule – SPT (Shortest Processing Time) – to calculate the priority value. The problem of using a single dispatching rule is that a certain rule may perform well in some situations but may perform very badly in some other cases. To overcome this shortcoming, a dispatching rule selection module has been applied in this agent-based system to automatically select dispatching rules according to the system status. This dispatching rule selection module has been realized by a Decision Tree model. This Decision Tree has been trained to select the best dispatching rule among SPT (Shortest Processing Time), LPT (Longest Processing Time), LWKR (Least Work Remaining), and MWKR (Most Work Remaining) based on three system status parameters: system utilization, average flow allowance, and percentage of unfinished jobs. For more information about the dispatching rules, please refer to Panwalkar and Iskander (1977). The three parameters are calculated as follows.

$$\text{system utilization} = \frac{numberOfWorkingMachines}{numberOfAllMachines}$$

$$\text{average flow allowance} = \frac{\sum \frac{jobDueDate - currentTime}{jobRemainingWork}}{numberOfUnfinishedJobs}$$

$$\text{percentage of unfinished jobs} = \frac{numberOfUnfinishedJobs}{numberOfAllJobsInOrder}$$

Before training the Decision Tree, a data set of 1000 records had been generated. Orders that had 2 to 10 letters had been randomly generated. For each order, the four dispatching rules (i.e. SPT, LPT, LWKR, and MWKR) had been respectively applied. Whenever a dispatching decision (i.e. sending a part shuttle to a machine) had been made, data that contained the name of the dispatching rule and values of the three system status parameters were recorded. When the order

was finished, the makespan for the order was recorded. To find the best dispatching rule for each order, only the data for a dispatching rule that had the minimum makespan were kept for training the Decision Tree. The training of the Decision Tree had been carried out in RapidMiner 8. The input fields for the Decision Tree were the three system parameters. The output of the Decision Tree was the selection of the dispatching rule. A test data set of 100 records had been generated to verify and the validate the Decision Tree embedded in the agent-based system. The prediction correctness rate of Decision Tree was achieved at 93%. The test cases also showed that the average makespan with the Decision Tree-based dispatching rule module was reduced compared to the application of any individual dispatching rule (i.e. SPT, LPT, LWKR, and MWKR).

It can be observed that the Agent-based scheduling system was improved by embedding a Decision Tree-based dispatching rule selection module. However, this system has been developed using specific software tools like JADE and RapidMiner. This makes it impossible for other manufacturers, who do not possess the tools, to make use of the developed system. To allow other industrial users to be able to access and make use of this computational platform, a KECM that combines the standardized Agent-based model and the standardized Decision Tree model must be developed. Moreover, these two standardized models must be combined, and this model combination must be formally represented to allow industrial users to access the two models as a whole. This is because the combined model serves the whole functionality of real-time scheduling; while the individual models cannot.

In the following sections, the standardized models for the Agent-based model and the Decision Tree model are developed, respectively. Then, the combined model is developed.

Finally, the utilization of the combined model in the real-time scheduling scenario described in the previous section is discussed. Since the technology selected to represent the KECM is OWL, all the models are represented in OWL.

5.2.3 Development of A Formal Representation for Agent-based Models

Figure 5.21 presents the representation of the developed standardized agent-based model. This model has been created based on the JADE Agent-based system. An agent is a computational module that inhabits an agent platform and typically offers one or more computational services (Bellifemine et al., 2007). The *AgentBasedModel* is the entity that represents the model of an Agent-based system. An *AgentBasedModel* should have at least one *Agent*. Each *Agent* must have an *Agent Identifier (AID)* for its notion of identity. Any parameter of the *Agent* is captured in *AgentParameter*. If the *AgentParameter* has a domain meaning, it can be connected to a domain concept (i.e. *owl:class*) through the *hasParameter* relationship. Any task that is carried out by an *Agent* is captured in *Behaviours*. Each *Behaviour* defines the general framework of a task. For example, a *Behaviour* can be categorized into *SimpleBehaviour* and *CompositeBehaviour*. *SimpleBehaviour* can be further classified into *OneShopBehaviour* that only executes once, *CyclicBehaviour* that executes repeatedly until a certain condition is matched, and *TickerBehaviour* that executes whenever a certain time passes, etc. The actual operation that needs to be carried out in a *Behaviour* is defined in an *Action*. Figure 5.21 demonstrates some examples of *Action*s like *CreateAgentAction*, *ReceiveMessageAction*, and *SendMessageAction*, etc. Another feature of an Agent-based system is the message

communication between agents. The *ACLMessage* (Agent Communication Language Message) represents the messages exchanged between agents. Each *ACLMessage* has a *Sender* and a *Receiver* which are *Agent*s. Any *ACLMessage* has a *CommunicativeAct* that captures the general function or action of the message. A *CommunicativeAct* is represented as a string like *CONFIRM*, *CFP*, and *INFORM*, etc. The dotted green arrows indicate the enumeration values of a data property. An *ACLMessage* also has a *MessageContent* which can be a string or an instance of an *owl:class*. For any *Action* related to a message communication like *SendMessageAction*, the corresponding *ACLMessage* should be connected to the *Action* through a *hasMessage* relationship.

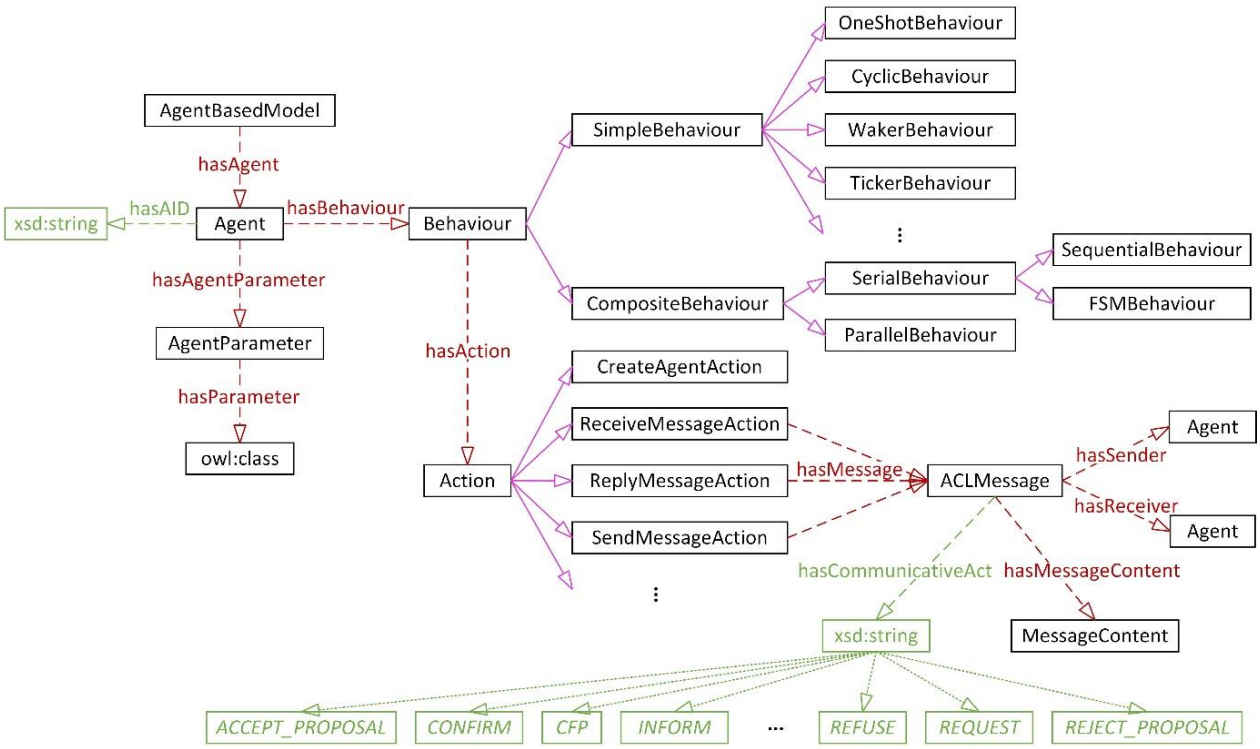


Figure 5.21 Representation of the Agent-based Model

5.2.4 Development of A Formal Representation for Decision Tree Models

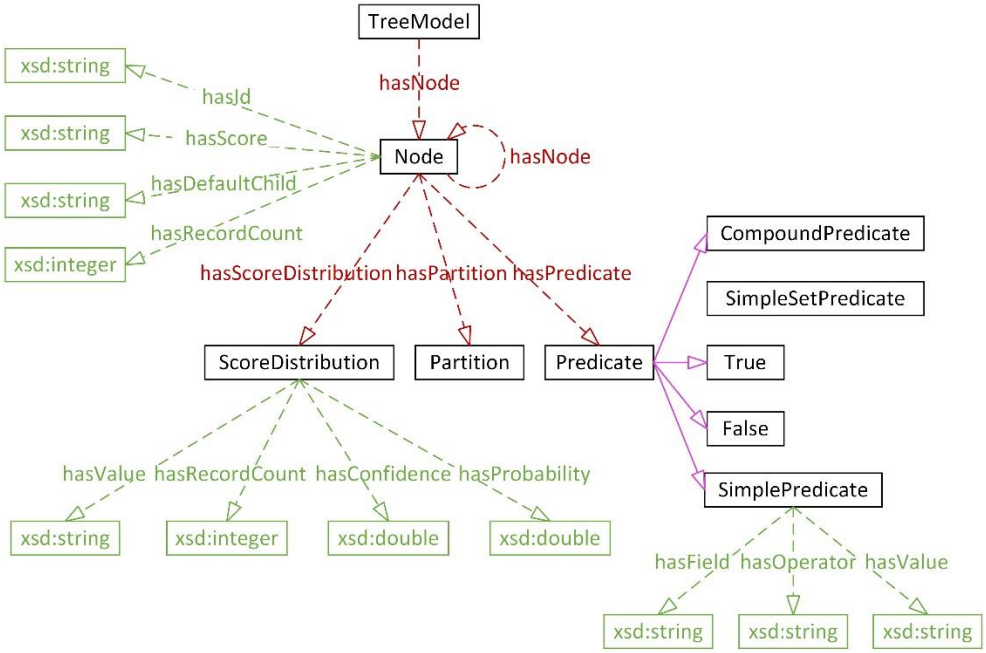


Figure 5.22 Representation of the Decision Tree model in OWL

Figure 5.22 presents a representation of the developed Decision Tree model. This model is developed based on the PMML – Tree Model (DMG, 2016). The entity names are borrowed from the PMML. Object properties have been added to better fit the Decision Tree model in the OWL language. The *TreeModel* represents the overall entity of a Decision Tree. The *Node* element is an encapsulation for either defining a split or a leaf in a tree model. Every *Node* has a *Predicate* that identifies a rule for choosing itself or any of its siblings. The *SimplePredicate* defines a rule in the form of a Boolean expression. The rule has attributes through *hasField*, *hasOperator*, and *hasValue* data properties. The *hasField* property captures the name of an input attribute of the *TreeModel*. The *hasOperator* property represents mathematical symbols like *equal*, *notEqual*, *lessThan*, *lessOrEqual*, *greaterThan*, or *greaterOrEqual*. The *hasValue* property

captures the value for the Boolean expression. A *ScoreDistribution* is an element of *Node* to represent segments of the score that a *Node* predicts in a classification. The *Partition* provides distribution information for all records for a *Node*. For more information about the entities of this model, please refer to (DMG, 2016).

5.2.5 Development of A Composed Agent-based and Decision Tree Model

In the previous sections, the individual computational models for the Agent-based model and the Decision Tree model have been developed. In this section, the model combination for these two models has been developed. Based on the previous definition of the combined computational model, a combined model is also a computational model. So, the combined computational models can be easily integrated into the KECM. Figure 5.23 demonstrates the KECM model for the case study.

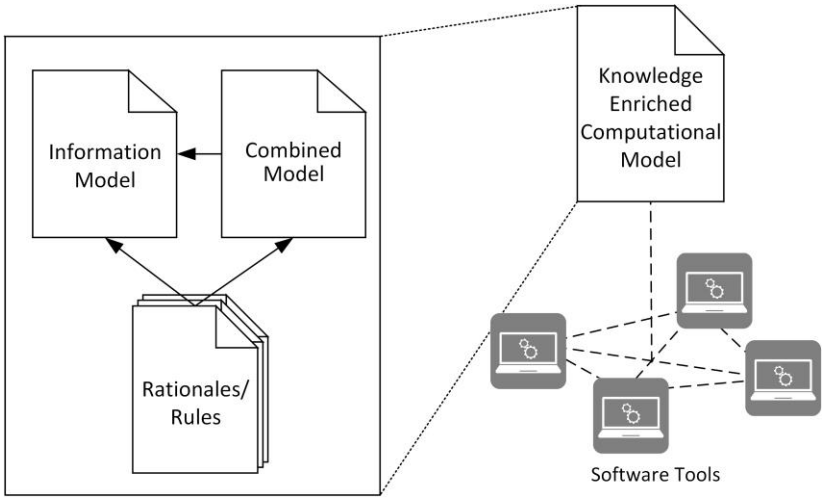


Figure 5.23 Development of the composed Agent-based and Decision Tree model

Figure 5.24 presents a representation of the implemented composed Agent-based and

Decision Tree model. The purple boxes represent the instances that are generated based on the defined classes introduced in the previous sections. In this case study, the output of the Decision Tree model is a string of the name of a dispatching rule selected like “SPT”, “LPT”, or “LWKR”; the input of the agent-based model from the Decision Tree model is an integer value. So, their input and output are not directly connected but are separately represented by their domain concept instances: *dispatchingRule_output* and *dispatchingRule_input*. The mappings between the name string of the dispatching rules and the integer values are SPT – 1, LPT – 2, LWKR – 3, and MWKR – 4.

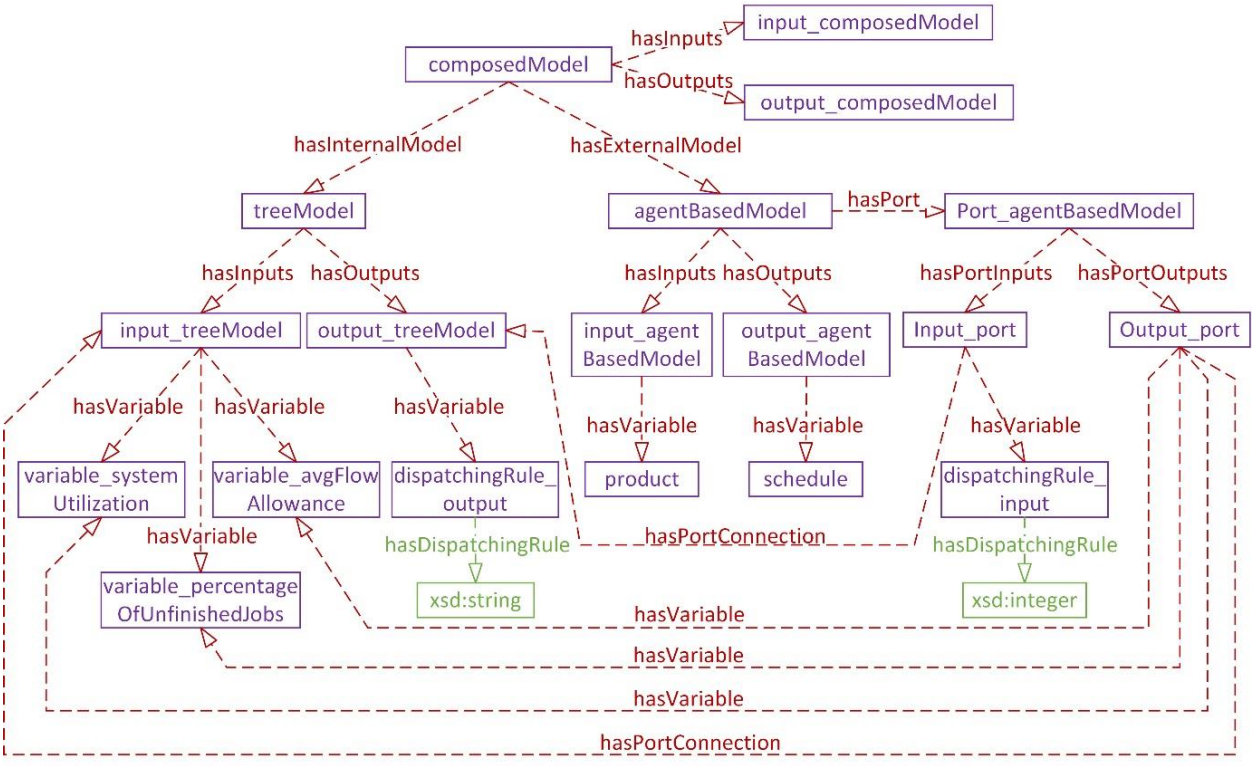


Figure 5.24 Representation of the implemented combined model

Rationales/rules have been developed to support this model combination. In this case study,

two types of rules have been created. First, rules have been developed to connect the input and output of the composed model to those of the external model (i.e. the Agent-based model). Second, rules have been developed to describe the mappings between a string output from the Decision Tree model and the integer port input of the Agent-based model. Likewise, all the rules are implemented in SWRL rules. An example of the rule of the first type is given as follows.

The meaning of this rule is: if the *agentBasedModel* has Inputs and the Inputs has a variable that has a manufacturing domain meaning, then the *composedModel* should have the same variable. An example of the second type of rule is given as follows.

Inputs(?inputs), hasInputs(composedModel, ?inputs), Inputs(?inputs_ex),
ComputationalModel(?computationalModel),
hasExternalModel(?composedModel, ?computationalModel),
hasInputs(?computationalModel, ?inputs_ex), ManufacturingConcept(?mc),
hasVariable(?inputs_ex, ?mc) -> hasVariable(?inputs, ?mc)

Outputs(?outputs), hasOutputs(treeModel, ?outputs), DispatchingRule(?dr1),
hasVariable(?outputs, ?dr1), xsd:string(?drName), swrlb:equal(?drName, "SPT"),
hasDispatchingRule(?dr1, ?drName), Port(?port), PortInputs(?portInputs),
hasPort(agentBasedModel, ?port), hasPortInputs(?port, ?portInputs), DispatchingRule(?dr2),
hasVariable(?portInputs, ?dr2) -> hasDispatchingRule(?dr2, 1)

The meaning of this rule is: if the output string of the *treeModel* equals “SPT”, then the integer value for the *portInputs* of the *agentBasedModel* should be 1. The screenshot of the implemented KECM, whose core is the composed model in protégé 5.2 is shown in Figure 5.25.

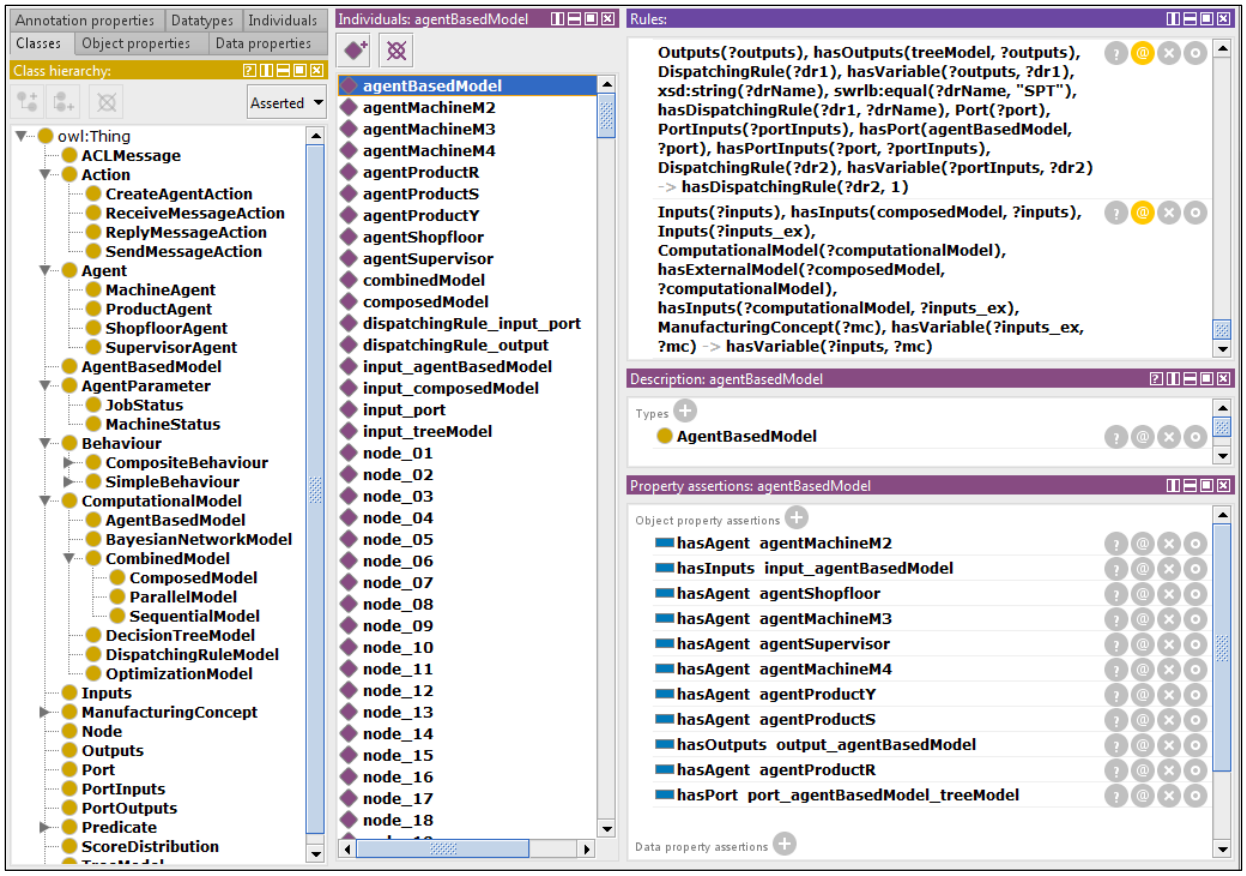


Figure 5.25 Screenshot of the implemented model combination in protégé 5.2

5.2.6 Utilization of the Composed Agent-based and Decision Tree Model

In this case study, the combined model developed in the last section has been used in the real-time scheduling scenario described in section 5.2. To support model combination, rationales/rules have been developed to automatically generate the input and output of the composed model and to automatically map the data of different types between the two models.

Figure 5.26 shows some results of reasoning the rationales/rules. The orange arrows indicate the corresponding rationale/rule that produced the reasoning results. The rules supporting the model combination can significantly enhance efficiency in manipulating the KECM model.

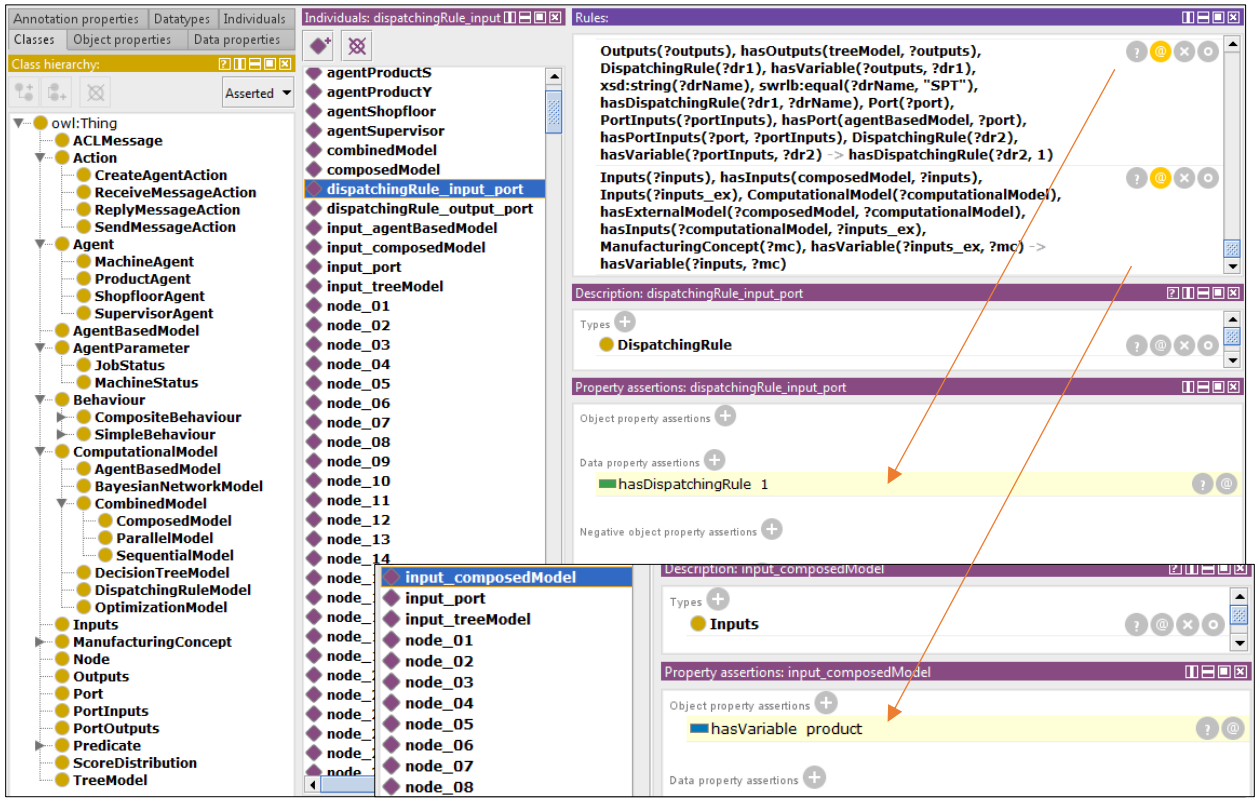


Figure 5.26 Using rationales to support model combinations

Currently, there are no free/open-source software tools (e.g., R, RapidMiner, and Knime, etc.) that can consume Decision Tree models represented in PMML. To further validate the idea of using the KECM to support model deployment, the consumption of the model composition has also been partially implemented. To consume the internal Decision Tree model, a parser that can process the Decision Tree model in OWL has been developed using the OWLAPI; and a code generator that can automatically generate Java code based on the Decision Tree model has been

developed. Figure 5.27 shows a screenshot of the code generator. It proves that the developed KECM can be easily consumed by computational platforms. The accessibility of computational models is also enabled, which is required by Smart Manufacturing. The plug-and-play capability of the computational models has been partially achieved by the proposed KECM.

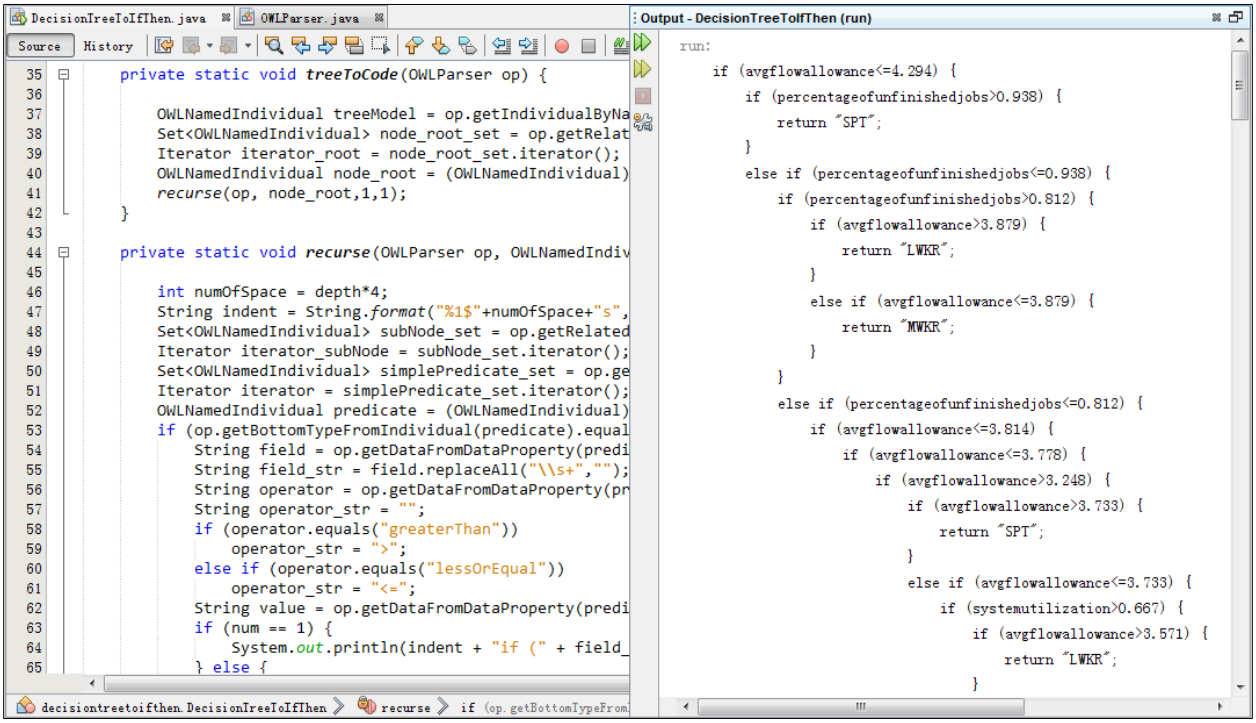


Figure 5.27 Screenshot of the code generator in Netbeans 8

CHAPTER 6. Utilization of the Knowledge Enriched Computational Model for Model Retrieval

In this chapter, an application of utilizing the KECM for model retrieval is introduced. Through modeling rationales to formally describe computational models, a semantics-based approach can be applied to measure the similarity between the semantic descriptions of the candidate computational models and that of the model retrieval requirements. In this chapter, the study of retrieving dispatching rule models is presented.

6.1 Introduction

Today, with the increasing complexity of industrial systems, researchers and industrial users do not want to build their computational models of industrial systems from scratch. An alternative approach is to seek for pieces of existing models to build their models and build complex systems by combining smaller sub-models (Henkel et al., 2010). To facilitate model reuse, the retrieval of models, which decides for potentially suitable models from a large number of available computational models becomes an important activity. It is important to rank the computational models based on their relatedness to the requirements given by the model user. Before the computational models can be deployed in an SM system, the ranked models should be selected and possibly combined to fulfill the user's requirements. Thus, a systematic approach to retrieve/select computational models and possibly combine models should be developed. In this chapter, a model retrieval and combination method, which conforms to the KECM, has been developed for retrieving dispatching rule models based on user-selected production objectives.

6.2 Model Retrieval with the Knowledge Enriched Computational Model

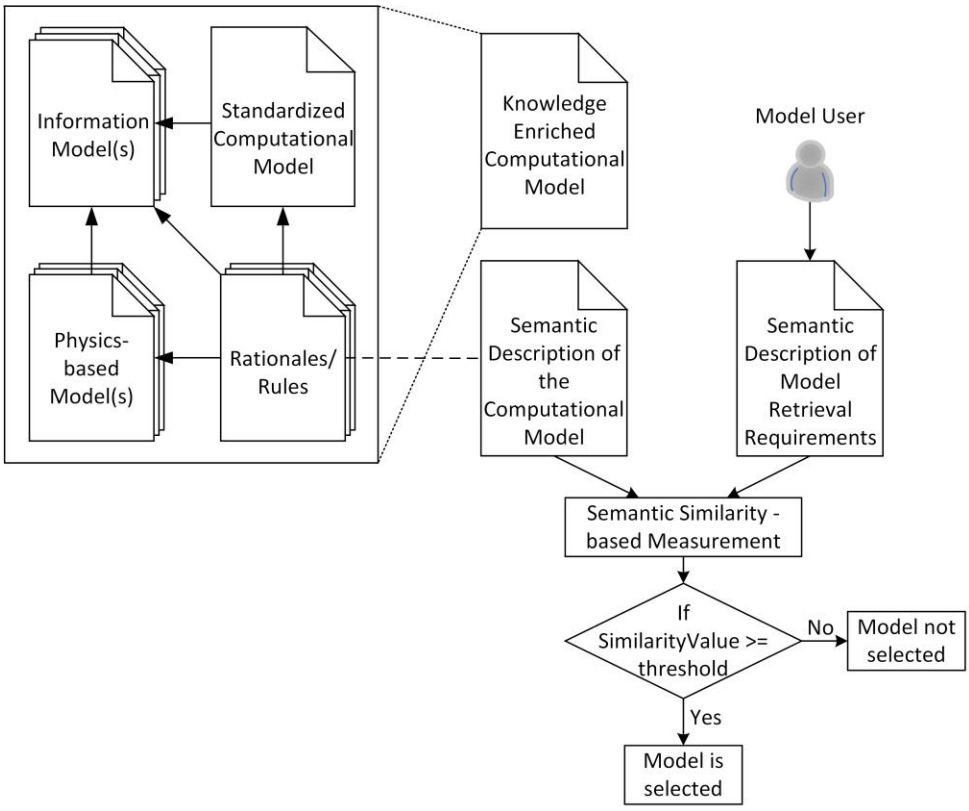


Figure 6.1 Retrieving computational models with the Knowledge Enriched Computational Model

Figure 6.1 presents a methodology to retrieve computational models with the proposed Knowledge Enriched Computational Model. To support model retrieval, rationales/rules are used to semantically describe the computational models. The model user, who intends to retrieve suitable models, provides the semantic description of the requirements for model retrieval. Through semantic similarity-based measurement, the semantic similarity values between the model retrieval requirements and the computational models can be calculated. If the similarity value is greater or equal to the threshold defined, the model can be selected. It is important to

note that the semantic description of the computational models and that of the model requirement should be defined in the similar fashion or structure. In this chapter, the study of retrieving dispatching rule models based on given production objectives is introduced.

6.3 Model Retrieval and Combination for Dispatching Rule Models

Literature about job shop scheduling, which studies how to appropriately allocate manufacturing resources to production tasks in traditional job shops, have been investigating better solutions for decades. Among various approaches (e.g., Branch and bound algorithm, meta-heuristics-based algorithms, and dispatching rules) used to solve the job shop scheduling problem, dispatching rules have been widely used in the industry. This is because they are easier to implement, and they yield reasonable solutions within a very short computational time. Normally, each one of the dispatching rules developed and utilized in today's scheduling systems only targets at one fixed production objective. To overcome the limitation of pursuing just one objective, combinations of dispatching rules, which combine two or more dispatching rules together, have been developed. But the combinations of dispatching rules are still fixed towards certain objectives, since either a single rule or a combination of rules, is pre-set by the scheduler. Thus, a lot of research effort has been made on the selection of dispatching rules with respect to three or four production objectives (Geiger et al., 2006; El-Bouri and Shah, 2006; Azadeh et al., 2012; Mouelhi-Chibani and Pierreval, 2010; Shiue, 2009; Baykasoglu et al., 2010; Scholz-Reiter et al., 2010; Heger et al., 2015; Chen et al., 2012; El-Bouri and Amin, 2015; Lin et al., 2008; Azadeh et al., 2015; Liu and Dong, 1996; Kızıl et al., 2006; Zhong et al., 2014; Shafiq et al.,

2010; Joseph and Sridharan, 2011; Kashfi and Javadi, 2015).

Recently, due to the highly competitive and globalized markets, manufacturers are facing the problem of constantly changing needs from a variety of customers. This requires manufacturers to acquire the ability to react fast and to adapt to the new customers' requests. Facing different customers, a manufacturer may have to achieve two or three production objectives at the same time. In the meantime, they need to manage their production resources as efficiently as possible. In the context of dispatching rule selection, this means suitable dispatching rules/combinations need to be selected/constructed for a user selected production objective or a combination of production objectives, especially for a randomly selected combination of objectives. However, the current dispatching rule studies are not sufficient to solve this problem. The current simulation-based or machine learning-based approaches have difficulties when facing new combinations of the objectives. This is because that to select dispatching rules, both simulation and machine learning-based approaches need to (1) enumerate the candidate individual dispatching rules and the combinations of the dispatching rules, and (2) collect data from executing the simulation models or from real production scheduling cases, and then (3) analyzing the simulation results or training the predictive model. However, these processes always require a lot of time, which makes it difficult to face the constantly changing needs of the customers.

A new approach that addresses the above-mentioned problems needs to be developed. In this dissertation, a novel semantics-based approach to retrieve a combination of dispatching rules given randomly selected combination of objectives has been proposed. Each of the dispatching rules and production objectives relates to a set of scheduling parameters like processing time,

remaining work, total work, job due date, operation due date, finish time, release date, tardiness, etc. These parameters are semantically interrelated. For example, tardiness is a quantity that measures the difference between a late job's finish time and its due date. Given a production objective that minimizes the total tardiness, it is better to finish jobs before their due dates. So, any dispatching rule that is related to prioritize jobs or operations with early due dates should be preferred. For a more complex objective that minimizes tardiness penalty, parameters like job's or operation's due date, finish time and job's late penalty should be considered at the same time. Here, there are 5 scheduling parameters in total (i.e. job due date, operation due date, job finish time, operation finish time, job's late penalty) related to the production objective through an "and" or an "or" relationship. When multiple production objectives are selected, it is even more important to sort out the interrelationships of the scheduling parameters. By formally defining all the scheduling parameters using semantic terms, all the production objectives and dispatching rules can be transformed into semantic expressions. Further, by comparing the formal semantic expressions between a production objective and each of the dispatching rules, dispatching rules that are more semantically similar can be selected to construct a combination of dispatching rules. With this idea, the semantic similarity-based approach can be put forward as a solution to measure the semantic similarity between the semantic expression of the production objectives and that of the dispatching rules (Zhang and Roy, 2018).

The semantics-based techniques originate from the exploration of the semantic web. Compared to the traditional web, the semantic web has enriched information (i.e. semantics) like class hierarchy, object properties, axioms, etc., which provides a formal description of concepts

and relationships within a given knowledge domain. Here, ontologies play a key role to define the precise vocabulary. A related technique - semantic similarity, which is also a measurement, defines the likeness between a set of concepts based on their semantic content, which is normally governed by an ontology (Harispe et al., 2017). In consideration of the dispatching rule selection problem, if the scheduling related concepts are formally defined in an ontology, semantic similarity technique can be applied to measure the similarity between a dispatching rule and a production objective (or a combination of production objectives). The similarity values can further be used for the selection of the suitable dispatching rules.

6.4 Problem Formalization

Each of the production objectives and the dispatching rules relates to one or more scheduling concepts, so a production objective (PO) and a dispatching rule (DR) can be represented as

$$PO \leftarrow f(P_1, P_2, \dots, P_m)$$

$$DR \leftarrow f(Q_1, Q_2, \dots, Q_n)$$

$$P_x, Q_y \in \{Scheduling\ Concepts\}$$

$$x \in 1,2, \dots, m; y \in 1,2, \dots, n$$

Where P_x or Q_y represents a scheduling concept; and *f* represents the logical combination of all the P_x that can describe a PO or a DR. The two basic logic combination types are AND and OR; and the combination can be mixed. Each concept P_x or Q_y can be further described by other concepts in a similar fashion as

$$P_x \leftarrow f(R_1, R_2, \dots, R_{m'})$$

$$Q_y \leftarrow f(S_1, S_2, \dots, S_{n'})$$

$$R_x, S_y \in \{Scheduling\ Concepts\}$$

$$x \in 1,2, \dots, m'; y \in 1,2, \dots, n'$$

Where R_x or S_y represents a scheduling concept that used to describe a P_x or a Q_y. Then the R_x or S_y can also be further defined by other concepts until all the concepts are well described. Thus, a PO or a DR is described by a set of semantic terms. For example, a PO and a DR can be described as

$$\text{PO} = \begin{Bmatrix} PO \leftarrow f(P_1, P_2) \\ P_1 \leftarrow f(R_1, R_2) \\ P_2 \leftarrow f(R_3) \\ R_1 \leftarrow f(T_1, T_2) \\ R_2 \leftarrow f(T_3) \\ R_3 \leftarrow f(T_4, T_5, T_6) \end{Bmatrix} \quad \text{DR} = \begin{Bmatrix} DR \leftarrow f(Q_1) \\ Q_1 \leftarrow f(S_1, S_2) \\ S_1 \leftarrow f(U_1, U_2) \\ S_2 \leftarrow f(U_3) \end{Bmatrix}$$

Where all the Ts and Us are the concepts do not need to be further described.

By formally capturing the scheduling concepts that are included in the above logical expressions, a semantic similarity between two single concepts can be directly calculated. Then the semantic similarity between a PO and a DR can be further evaluated by calculating the similarities between their semantic expressions. In order to identify the concepts, this paper proposes an extended Sustainable Manufacturing Ontology which captures the scheduling related concepts and relationships. Then the semantic expressions of all the production objectives and the dispatching rules can be presented using the semantic terms identified in the ontology. A tree matching based algorithm is proposed next to calculate the semantic similarity between the set of logical expressions of a PO and that of a DR. Finally, a way of generating the proper dispatching rule for a given production objective is described.

6.5 A Semantics-based Methodology for Dispatching Rule Selection

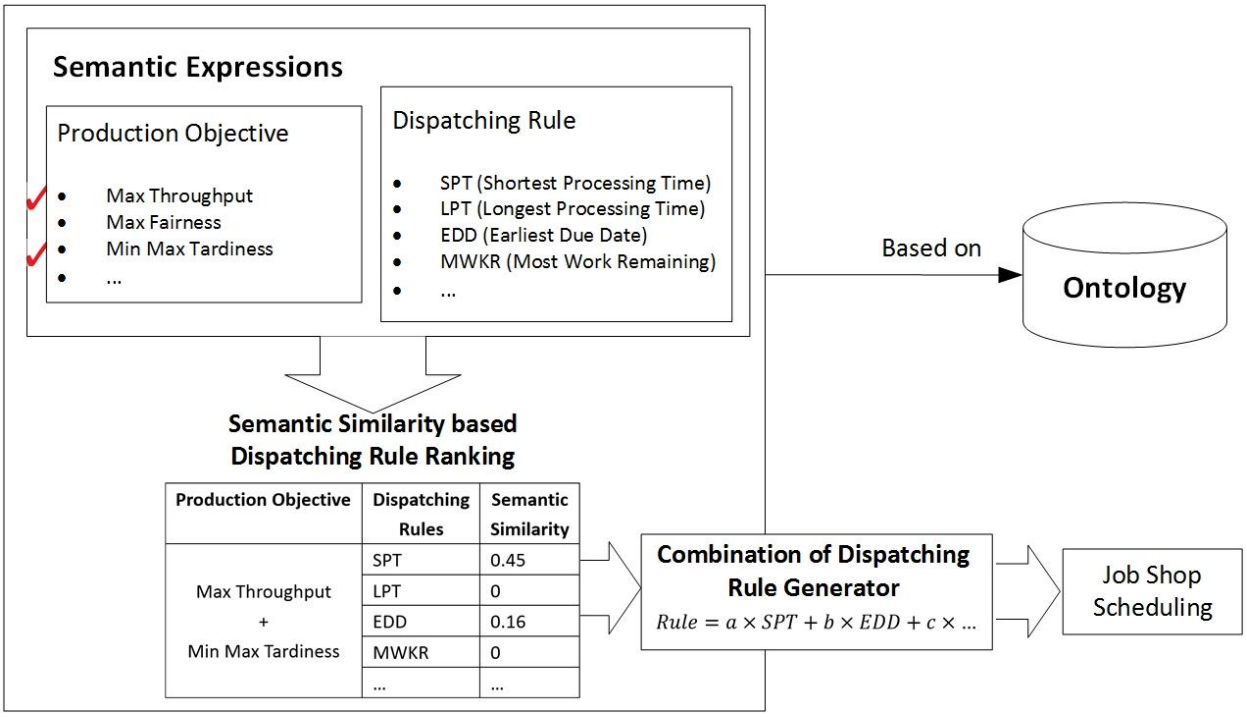


Figure 6.2 The semantics-based methodology for dispatching rule selection

Figure 6.2 demonstrates the proposed semantics-based methodology to solve the dispatching rule selection problem. The framework has the selected production objectives from the user as system inputs and a single dispatching rule/combination of dispatching rules which can be used directly in the job shop scheduling as an output. The proposed system includes four parts: (1) an ontology, (2) the semantic expressions of the production objectives and the dispatching rules, (3) a semantic similarity based dispatching rule ranking system, and (4) a generator to construct a combination of dispatching rules. The ontology defines basic manufacturing concepts, scheduling concepts and the relationship between concepts. The semantic expression of each production objective or each dispatching rule is defined using the concepts from the ontology. So,

this ontology serves as a concept repository to provide the semantic expressions of the production objectives and the dispatching rules with basic semantic terms. After the production objectives (one or more) have been selected by the user, all the dispatching rules will be ranked based on the semantic similarity values obtained by comparing the semantic expressions between the combination of the production objectives and each dispatching rule. The similarity values obtained will then be used to calculate the weights for each single dispatching rule to generate the final dispatching rule combination. The detailed descriptions about each part will be provided in the following sections.

This methodology conforms to the overall knowledge integration framework. All the semantic expressions for the dispatching rules and the production objectives can be captured into the rationales/rules in the overall framework. These rationales/rules that describe the domain meanings of the dispatching rules and the production objectives can be defined during model development. The underlying ontology that needed by the proposed semantics-based approach is the information model(s) that captured by the overall framework.

6.5.1 Sustainable Manufacturing Ontology

This section introduces the ontology that defines all the scheduling related concepts and their relations. The proposed ontology is based on one earlier work reported in Zhang et al. (2015). The earlier information model has been extended to include information that relates to job shop scheduling. Figure 6.3 presents a UML Class Diagram that represents the extended Sustainable Manufacturing Ontology. Concepts represented in black boxes are defined in the original

information model. The concepts represented in blue boxes are the extended concepts with respect to job shop scheduling domain. A concept has been added to the SMO: *AdministrativeEntity*. This concept represents an abstract entity of all the administrative concepts. Figure 6.4 expands the *AdministrativeEntity* concept into a hierarchical tree.

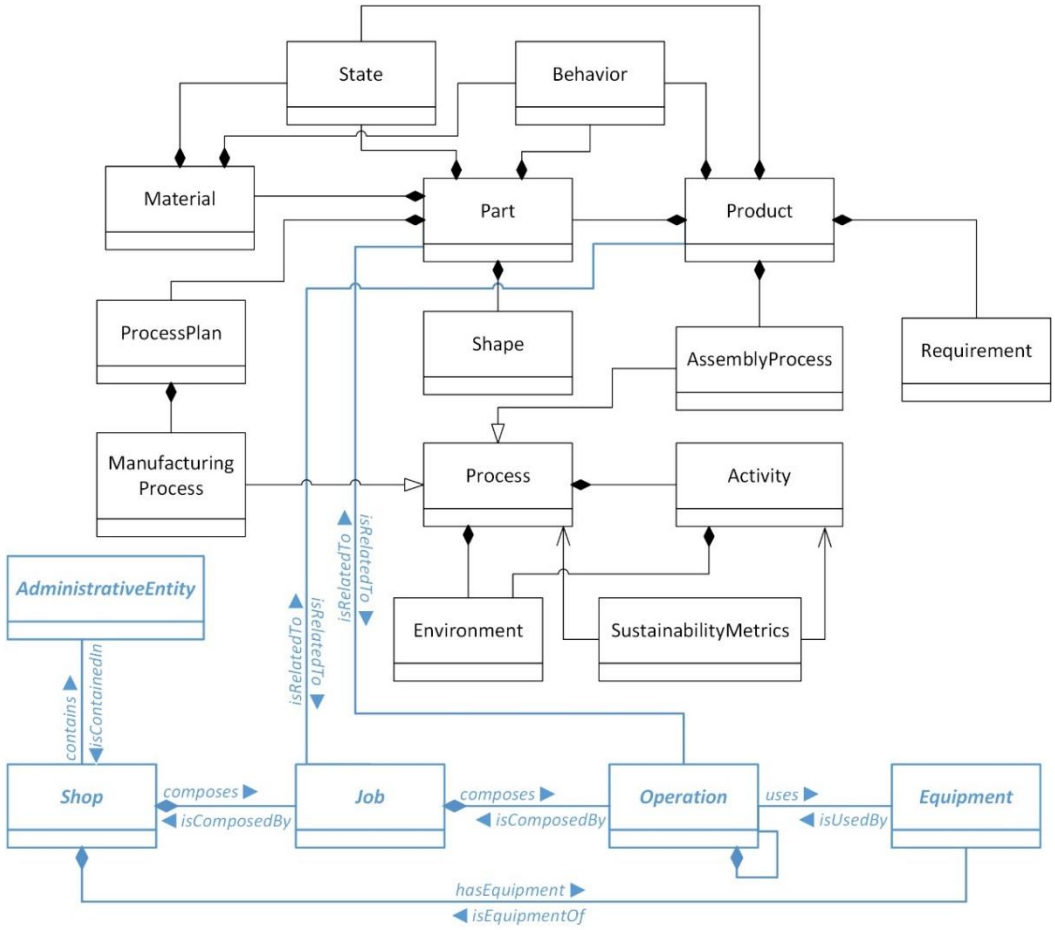


Figure 6.3 UML class diagram for the extended Sustainable Manufacturing Ontology (SMO)

For more information about the SMO, please refer to section 4.4.1.

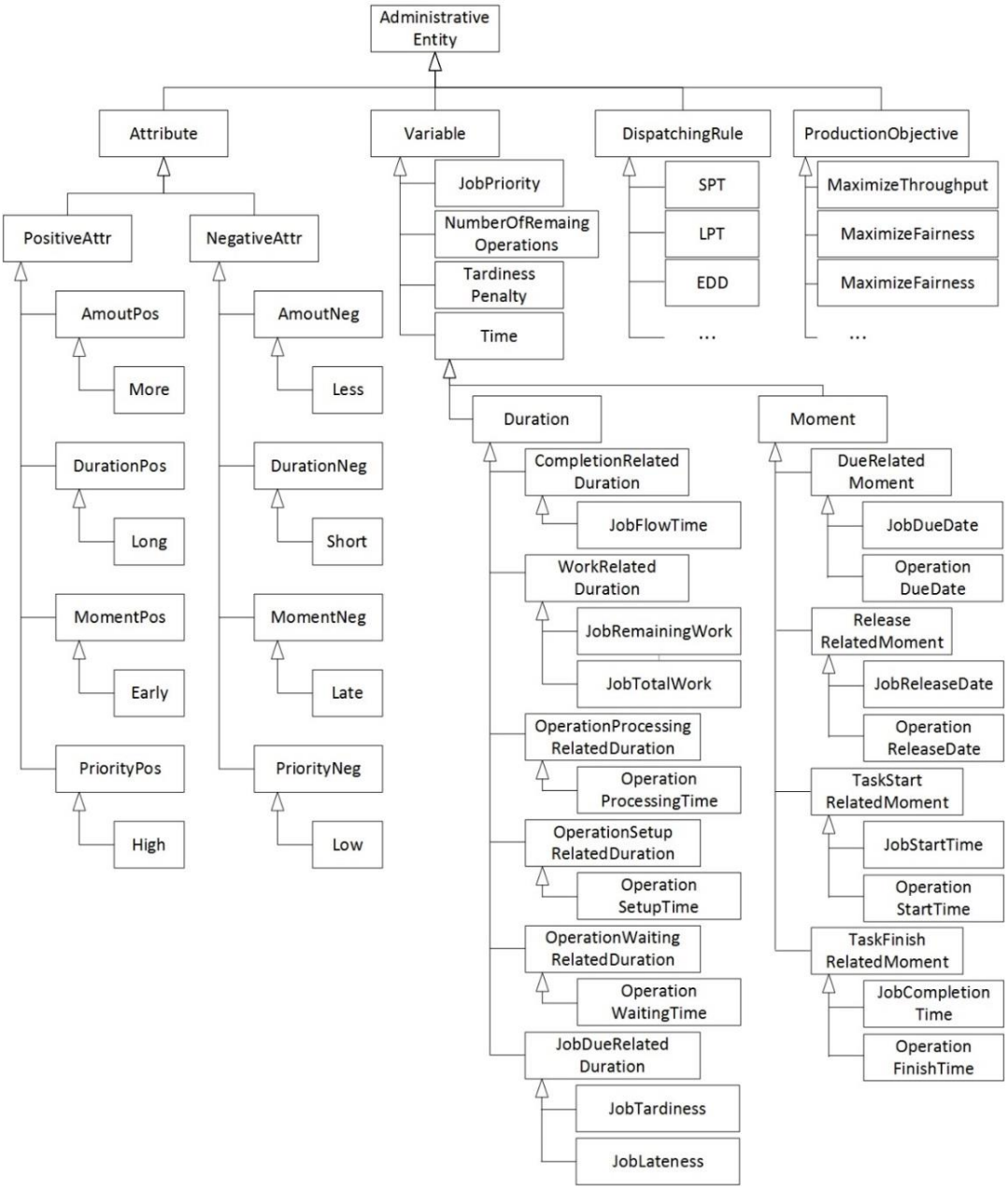


Figure 6.4 Hierarchical tree for *AdministrativeEntity*

6.5.2 Semantic Expressions of Production Objectives and Dispatching Rules

After the ontology has been introduced, the semantic expressions of the production objectives and the dispatching rules are presented in this section. In this work, we have studied and explored the semantic expressions for 10 production objectives and 16 dispatching rules.

These semantic expressions are written in the Manchester OWL Syntax (W3C Working Group, 2012).

The semantic expressions for the production objectives are listed below.

- Maximize Fairness

```
ProductionObjective
  and (relatesTo some (Job
    and (hasTime some (JobReleaseDate
      and (hasAttr only Early)))))
```

- Minimize Response Time

```
ProductionObjective
  and (relatesTo some (Job
    and (hasTime some (JobReleaseDate
      and (hasAttr only Late)))))
```

- Maximize Priority Conformity

```
ProductionObjective
  and (relatesTo some (Job
    and (hasWeight some (JobWeight
      and (hasAttr only High)))))
```

- Minimize Waiting Time Variance

```
ProductionObjective
  and (relatesTo some (Operation
    and (hasTime some (OperationWaitingTime
      and (hasAttr only Long)))))
```

- Maximize Throughput

```
ProductionObjective
  and (relatesTo some (Operation
    and (hasTime some (OperationProcessingTime
      and (hasAttr only Short)))))
```

- Minimize Tardiness

```
ProductionObjective
  and ((relatesTo some (Job
    and (hasTime some (JobDueDate
      and (hasAttr only Early)))
  or (relatesTo some (Operation
    and (hasTime some (OperationDueDate
      and (hasAttr only Early))))))
```

- Minimize Job Lateness Variance

```
ProductionObjective
  and (relatesTo some (Operation
    and (hasTime some (OperationDueDate
      and (hasAttr only Early)))))
```

- Minimize Setup Time

```
ProductionObjective
  and (relatesTo some (Operation
    and (hasTime some (OperationSetupTime
      and (hasAttr only Short)))))
```

- Minimize Tardiness Penalty

```
ProductionObjective
  and ((relatesTo some (Job
    and ((hasPenalty some (TardinessPenalty
      and (hasAttr only More)))
    or (hasTime some (JobDueDate
      and (hasAttr only Early))))))
  or (relatesTo some (Operation
    and (hasTime some (OperationDueDate
      and (hasAttr only Early))))))
```

- Minimize Makespan

```
ProductionObjective
  and (relatesTo some (Job
    and ((hasTime some (JobRemainingWork
      and (hasAttr only Long)))
    or (hasTime some (NumberOfRemainingOperations
      and (hasAttr only More))))))
```

The semantic expressions for the dispatching rules are listed below.

- 1/C
  (C: Tardiness penalty)

```
DispatchingRule
  and (prioritizes some (Job
    and ((hasPenalty some (TardinessPenalty
      and (hasAttr only More)))
    or (hasTime some (JobDueDate
      and (hasAttr only Early))))))
```

- EDD
  (Earliest Due Date)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobDueDate
      and (hasAttr only Early)))
    and (hasTime some (JobRemainingWork
      and (hasAttr only Long)))))
```

- MST
  (Minimum Slack Time)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobDueDate
      and (hasAttr only Early)))
      and (hasTime some (JobReleaseDate
        and (hasAttr only Late)))
      and (hasTime some (JobRemainingWork
        and (hasAttr only Long)))))
```

- OSL
  (Operation Slack)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationDueDate
      and (hasAttr only Early)))
      and (hasTime some (OperationProcessingTime
        and (hasAttr only Long)))
      and (hasTime some (OperationReleaseDate
        and (hasAttr only Late)))))
```

- FCFS
  (First Come First Serve)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobReleaseDate
      and (hasAttr only Early)))))
```

- MWKR
  (Most Work Remaing)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobRemainingWork
      and (hasAttr only Long)))))
```

- FOPNR
  (Fewest Operation
  Number Remaining)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasNumber some (NumberOfRemainingOperations
      and (hasAttr only Less)))))
```

- ODD (Operation Due Date)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationDueDate
      and (hasAttr only Early)))))
```

- LCFS (Last Come First Serve)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobReleaseDate
      and (hasAttr only Late)))))
```

- MOPNR (Most Operation Number Remaining)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasNumber some (NumberOfRemainingOperations
      and (hasAttr only More)))))
```

- LPT (Longest Processing Time)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationProcessingTime
      and (hasAttr only Long)))))
```

- SPT (Shortest Processing Time)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationProcessingTime
      and (hasAttr only Short)))))
```

- LWKR (Least Work Remaining)

```
DispatchingRule
  and (prioritizes some (Job
    and (hasTime some (JobRemainingWork
      and (hasAttr only Short)))))
```

- SST (Shortest Setup Time)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationSetupTime
      and (hasAttr only Short)))))
```

- LWT (Longest Waiting Time)

```
DispatchingRule
  and (prioritizes some (Operation
    and (hasTime some (OperationWaitingTime
      and (hasAttr only Long)))))
```

- W (Weight) DispatchingRule and (prioritizes some (Job and (hasWeight some (JobWeight and (hasAttr only High)))))

The semantic expression of each production objective is created based on its related scheduling parameters to best describe the objective. The descriptions of each production objective are given as follows:

- Maximize Fairness: In the absence of guidance from the user or other performance related guidance, all the jobs should be treated equally, which means jobs should be sequenced in a "first come first serve" order;
- Maximize Priority Conformity: When the priorities of jobs are assigned, scheduling should favor the jobs with higher priority;
- Maximize Throughput: Maximizing the number of finished operations for a given length of time;
- Minimize Job Lateness Variance: Balance the lateness of all the jobs. The case of finishing some jobs early but having several very late jobs is not preferred.
- Minimize the Makespan: Minimizing the total time length of the schedule;
- Minimize Tardiness: Minimizing the tardiness of jobs by focusing on the due dates;
- Minimize Response Time: The scheduling discipline should attempt to achieve low response time by processing the newly released jobs as soon as possible;
- Minimize Setup Time: Focusing on minimizing the setup time;
- Minimize Tardiness Penalty: Considering due dates and the tardiness penalty in scheduling;

- Minimize Waiting Time Variance: Waiting should be fair too. Minimizing the difference between the operation waiting time.

The semantic expression of each dispatching rule is, however, described based on how the dispatching rule should be applied in scheduling. The detailed discussion about each dispatching rule can be found in Panwalkar and Iskander (1977) and Haupt (1989).

When multiple production objectives are selected, the semantic expression should be constructed by combining the semantic expression of each individual objective through an "and" relationship. As an example, the semantic expression of combining the "Maximize fairness" objective and the "Minimize Tardiness Penalty" objective is presented below.

- Maximize Fairness + Minimize Tardiness Penalty

```
ProductionObjective
and ((relatesTo some (Job
  and (hasTime some (JobReleaseDate
    and (hasAttr only Early)))))
and ((relatesTo some (Job
  and ((hasPenalty some (TardinessPenalty
    and (hasAttr only More)))
  or (hasTime some (JobDueDate
    and (hasAttr only Early))))))
or (relatesTo some (Operation
  and (hasTime some (OperationDueDate
    and (hasAttr only Early)))))))
```

6.5.3 Semantic Similarity Measurement

With the semantic definitions of all the production objectives and dispatching rules, semantic similarity measurement can then be carried out to calculate the similarity between the user-selected production objectives (one or more) and each of the dispatching rules. In order to do this, a tree structure of the semantic expression will be discussed; a tree matching based

algorithm and a semantic similarity measurement are introduced in this section. The semantic similarity values (output of the measurement) can then be used for generating the combination of dispatching rules.

6.5.3.1 The Tree Structure of The Semantic Expressions

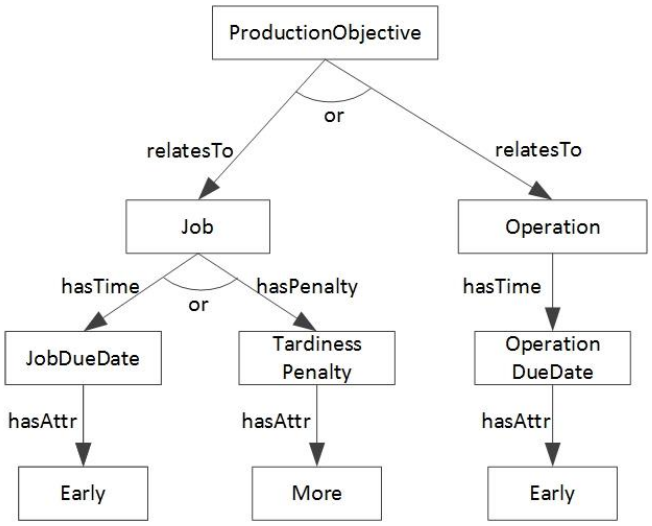


Figure 6.5 Tree structure for the "Minimize Tardiness Penalty" objective

The semantic expression of each production objective or dispatching rule written in the Manchester OWL syntax forms a tree structure: each concept can be represented as a node; each relationship can be represented as an edge, and the relationships between all the edges leaving from the same node are represented as the "and/or" relationships. Figure 6.5 presents the tree structures for a single objective "Minimize the Tardiness Penalty" and Figure 6.6 presents the structure of a combined objective "Maximize Fairness + Minimize Tardiness Penalty". So, the comparison between the semantic expressions can be transformed to the comparison between the two trees which contain the semantic nodes. Next section illustrates a tree matching based

algorithm for measuring semantic similarity.

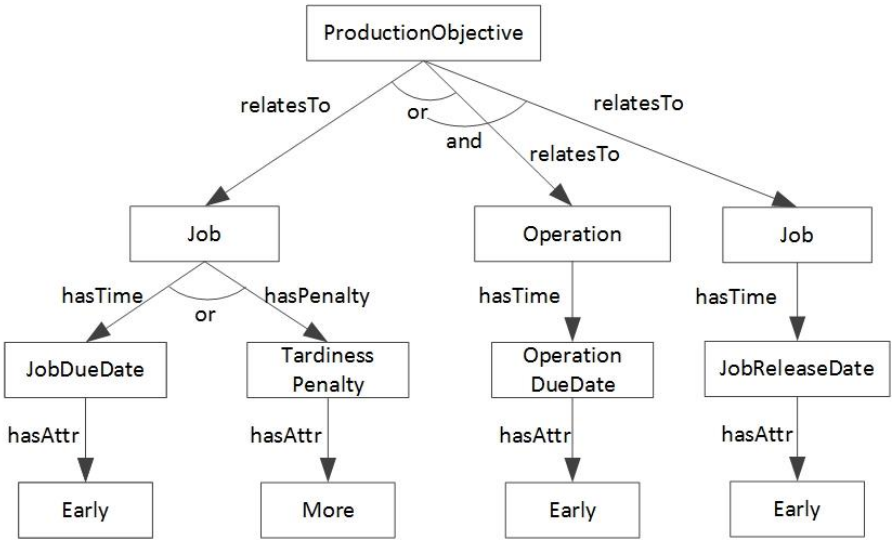


Figure 6.6 Tree structure for a combined "Minimize Tardiness Penalty" and "Maximize Fairness" objective

6.5.3.2 Tree Matching Based Algorithm for Semantic Similarity

A tree matching based algorithm has been developed to calculate the semantic similarity between a dispatching rule tree and a production objective tree. The algorithm starts from the roots of the two trees. The nodes from a dispatching rule's tree are compared with the nodes from a production objective's tree from the top layer to the bottom layer. All the trees have three layers from top to bottom: (1) a dispatching rule/production objective layer, (2) a job/operation layer, (3) a scheduling parameters layer and (4) an attributes layer. The similarity between the two whole trees can then be calculated by summing up the similarities of all the layers. However, the similarities of different layers should not have the same weight. A similarity between two concepts which have higher positions on the tree should have larger weights than the ones have

lower positions. The reason is that the concepts in the upper layers are more general and are more important than the ones in the lower layers. On the contrary, concepts in the lower layers include more specific and detailed information so that they have less influence on the similarity measurement. So, based on this idea, the weight of different layers has been developed as follows:

$$\mu_l = \frac{\frac{1}{depth(l)+1}}{\frac{1}{depth(l_0)+1} + \frac{1}{depth(l_1)+1} + \cdots + \frac{1}{depth(l_k)+1}}$$

Where *l* represents the current layer, $depth(l)$ is the depth of the *l*th layer in the tree. l_0 is the root layer which has $depth(l_0) = 0$. l_k is the deepest layer. The increment of the depth between each two layers is 1. It can be observed that $\mu(l_0) + \mu(l_1) + \cdots + \mu(l_k) = 1$. The final similarity between the two trees will then be given by

$$sim_{final} = \mu_1 \times sim_{l_1} + \mu_2 \times sim_{l_2} + \cdots + \mu_k \times sim_{l_k}$$

Where sim_{l_x} is the similarity value of layer x.

When the algorithm traverses the trees, not all the branches will be visited. For the edges leaving from the same node, if their relationship is an “and”, then all these edges will be visited; if their relationship is an “or”, then only one edge will be visited (The edge which has the highest similarity valued ending node will be visited). Figure 6.7 shows an example of the tree traversal strategy. Next section will present the detail of how to calculate the semantic similarity within a layer.

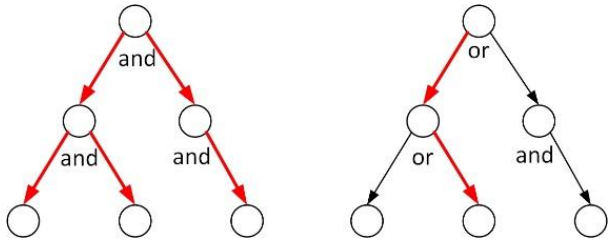


Figure 6.7 Tree traversal strategy

Table 6.1 Threshold values

Number of Objectives Combined	Threshold Values
1	0.8
2	0.7
3	0.5

```
function calculateSemanticSimilarity {
    sim = 0;
    maxDepth = getMaxDepth(ruleTree);
    for (i = 0; i < depth of the dispatching rule tree; i++) {
        layerWeight = calculateLayerWeight(i, maxDepth);
        layerSim = calculateLayerSimilarity(ruleTree, ObjTree, i);
        if layerSim >= threshold {
            sim = sim + layerWeight * layerSim;
        } else {
            Break;}
    }
    return sim;}
```

Figure 6.8 Pseudo codes for the calculateSemanticSimilarity function

```
function calculateLayerSimilarity(ruleTree, ObjTree, depth) {
    layerSim = 0;
    objNodeList = objTree.getVisitNodesInLayer(depth);
    ruleNodeList = ruleTree.getVisitNodesInLayer(depth);
    if size of objNodeList >= size of ruleNodeList {
        for objNode in objNodeList {
            simMax = 0;
            for ruleNode in ruleNodeList {
                nodeSim = calculateNodeSimilarity(objNode, ruleNode);
                if (nodeSim > simMax) {
                    simMax = simNode; }
            } layerSim = layerSim + simMax/size of objNodeList; }
    } else {
        for ruleNode in ruleNodeList {
            simMax = 0;
            for objNode in objNodeList {
                nodeSim = calculateNodeSimilarity(objNode, ruleNode);
                if (nodeSim > simMax) {
                    simMax = simNode; }
            } layerSim = layerSim + simMax/size of ruleNodeList;
        }
    } Return layerSim;
}
```

Figure 6.9 Pseudo codes for the calculateLayerSimilarity function

The traversal of the trees will stop either when it reaches the bottom layer, or the similarity value of a certain layer is too low so that there is no necessity to go to the next layer. Threshold values have been defined to govern the stop of the traverse. The thresholds are configured with a sensitivity analysis of the semantic similarity values (will be discussed in section 6.6.3). The definition of the threshold relates to the number of the objectives combined. When more objectives are combined, the semantic measure in a certain layer includes more concepts, which can lead to lower semantic similarity value. So, the threshold value is lower when more objectives are combined. The threshold values are shown in Table 6.1. If the semantic similarity value falls below the threshold, the tree traversal will be stopped.

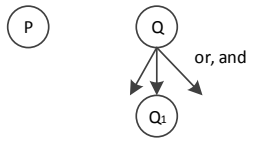
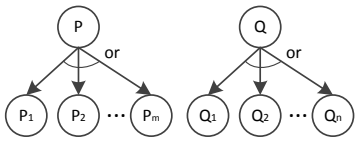
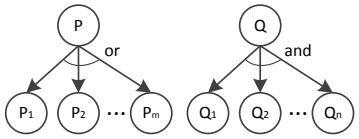
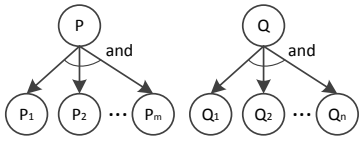
The pseudo code of the tree matching algorithm is provided in Figure 6.8 and Figure 6.9 to illustrate the process. In the pseudo code, the *getMaxDepth* function calculates the maximum depth of the trees. The *calculateLayerWeight* function calculates the layer weight based on the equation given above. The function *calculateNodeSimilarity* is described in the next section.

6.5.3.3 Tree-based Semantic Similarity Measurement

The semantic similarity measurement between the nodes in one layer is presented in Table 6.2. The semantic similarity between every two nodes is calculated based on the similarity of their children nodes. The $sim(P, Q)$ in the table represents the semantic similarity between the two concepts P and Q. It is a value that ranges from 0 to 1, in which 0 means the two concepts are totally different concepts and 1 means they are the same concept. In this paper, the similarity measurement between every two nodes is achieved through an edge-based semantic similarity

measurement proposed by Wu and Palmer (1994).

Table 6.2 Semantic similarity within a layer

Relation	Graph Presentation	Semantic Similarity Measurement
None - any		$sim_{none}(P,Q) = 0$
or - or		$sim_{oror}(P,Q) =$ $Max\begin{Bmatrix} sim(P_1,Q_1), sim(P_1,Q_2), \dots, sim(P_1,Q_n), \\ sim(P_2,Q_1), sim(P_2,Q_2), \dots, sim(P_2,Q_n), \\ \dots \\ \dots \\ sim(P_m,Q_1), sim(P_m,Q_2), \dots, sim(P_m,Q_n) \end{Bmatrix}$
or - and		$sim_{orand}(P,Q)$ $= Max\begin{Bmatrix} sim_{onlyand}(P_1,Q), sim_{onlyand}(P_2,Q), \dots \\ \dots, sim_{onlyand}(P_m,Q) \end{Bmatrix}$
and - and		If m > n, $sim_{\text{andand}}(P,Q)$ $Max\{sim(P_1,Q_1) + sim(P_1,Q_2) + \cdots + sim(P_1,Q_n)\} +$ $Max\{sim(P_2,Q_1) + sim(P_2,Q_2) + \cdots + sim(P_2,Q_n)\} +$ $\dots$ $= \frac{Max\{sim(P_m,Q_1) + sim(P_m,Q_2) + \cdots + sim(P_m,Q_n)\}}{m}$ Else, $sim_{\text{andand}}(P,Q)$ $Max\{sim(Q_1,P_1) + sim(Q_1,P_2) + \cdots + sim(Q_1,P_n)\} +$ $Max\{sim(Q_2,P_1) + sim(Q_2,P_2) + \cdots + sim(Q_2,P_n)\} +$ $\dots$ $= \frac{Max\{sim(Q_m,P_1) + sim(Q_m,P_2) + \cdots + sim(Q_m,P_n)\}}{n}$

The Wu and Palmer's measurement is utilized to calculate the semantic similarity between the two concepts on the extended Sustainable Manufacturing Ontology's taxonomy hierarchy. Wu and Palmer's method considered the position relation of two concepts *P* and *Q* to their nearest common ancestor *C* to calculate similarity. Here, *C* is located at the lowest position on the ontology hierarchy among all the common ancestors of *P* and *Q*. The mathematical formula for calculating the similarity between *P* and *Q* is given by

$$sim(P,Q) = \frac{2H}{D_p + D_q + 2H}$$

where D_p and D_q are the minimum edge (is-a edge) counts from *P* to *C* and *Q* to *C* respectively. *H* is the minimum edge count from *C* to the root node of the ontology.

6.5.4 Combination of Dispatching Rules Generation

Once the semantic similarity between the user-selected production objectives and each of the dispatching rules have been calculated, dispatching rules that have higher similarity values will be selected for generating the combination of dispatching rules. This combination of dispatching rules can then be used directly in job shop scheduling. Dispatching rules with the similarity values greater than a threshold will be selected. This threshold is defined as same as the threshold for traversing the tree (Table 6.1). Then the combination of dispatching rules can be formed as follows:

$$f = \rho_1 Rank(rule_1) + \rho_2 Rank(rule_2) + \cdots + \rho_u Rank(rule_u)$$

$$\rho_i = \frac{sim_i}{sim_1 + sim_2 + \cdots + sim_u}$$

Where f represents the ranking value of a job; the smaller this value is, the more priority this job has. $Rank(rule_i)$ returns a ranking value of the job according to the priority value given by $rule_i$. ρ_i is the coefficient associated with each dispatching rule. A dispatching rule that has higher similarity value has higher ρ_i value.

6.6 Verification and Results

As a proof-of-concept, this section provides a verification of the proposed semantics based dispatching rule selection approach. The implementation has been developed and the dispatching rule selection results of the implementation have been compared to their performances from simulation. The next two sections give a brief introduction to the implementation of the proposed approach and the simulation-based experiment.

6.6.1 Implementation

Figure 6.10 demonstrates the architecture of the implementation. Figure 6.11 shows the screenshot of the implemented extend Sustainable Manufacturing Ontology in protégé (BMIR, 2018) The proposed semantic expressions of the dispatching rules and production objectives are implemented as an “Equivalent To” class in protégé. Apache Jena (2018), which is an open source Java framework for semantic web related applications, has been used to access the OWL file generated from protégé. The application of semantics-based dispatching rule selection including the semantic similarity measurement and generation of the combination of dispatching rules is programmed in Java on top of Apache Jena.

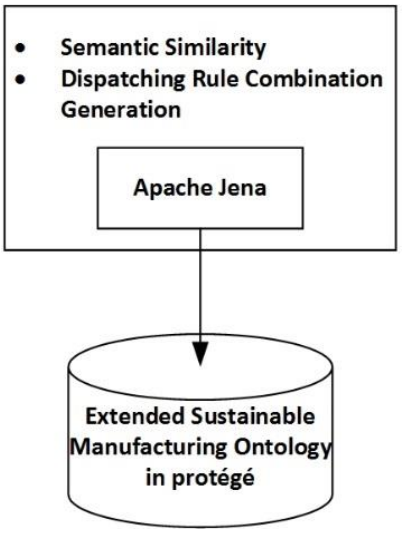


Figure 6.10 Architecture of the implementation

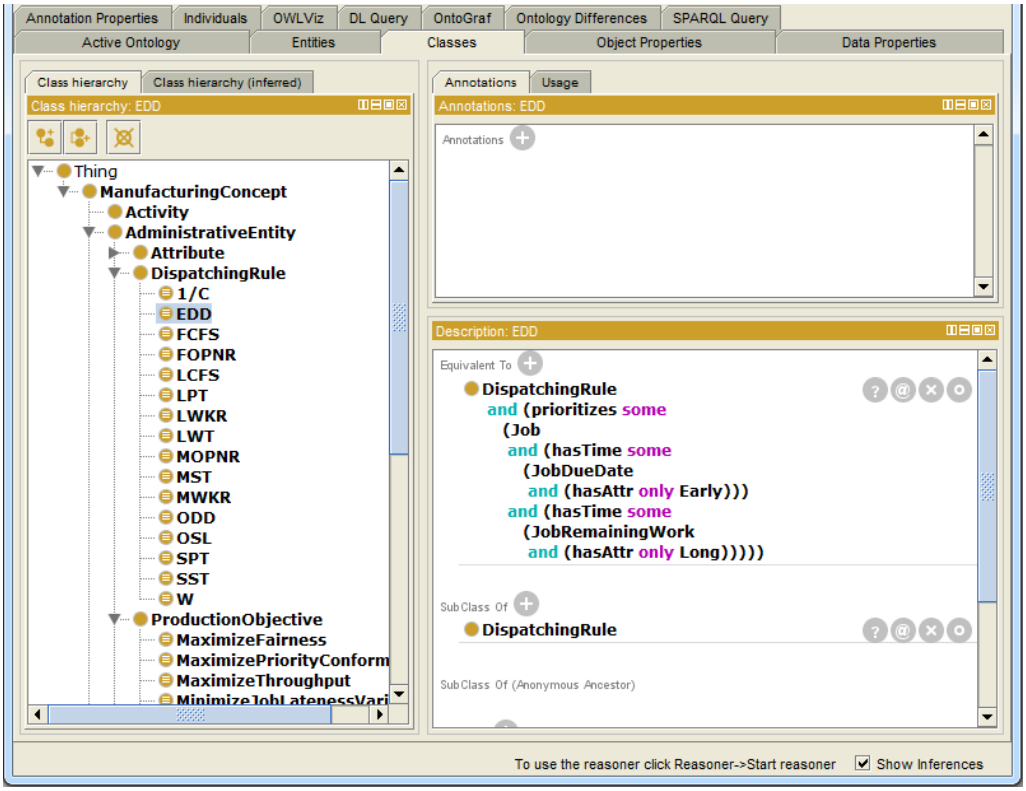


Figure 6.11 Implementation of the Sustainable Manufacturing Ontology and the implemented semantic expressions

6.6.2 Simulation-Based Experiment

A simulation program for dispatching rule-based scheduling has been developed to test the performance of the selected dispatching rules. Job shop scheduling problems with 8 machines and 50 jobs are randomly generated. Wilbrecht and Prescott (1969) have demonstrated that 6 machines were adequate to represent the complex structure of a job shop. The number of operations for each job is uniformly distributed between 1 and 8. The processing times are drawn from a uniform distribution between 3 and 10. The release date is uniformly distributed between 0 to 100. The due dates of jobs are assigned based on the method of total work-content (TWK) (Blackstone et al., 1982). In this study, the due date of a job is set at 5 times of a random number which is between its TWK and 3 times of its TWK. The due date setting can be represented formally as follows:

$$\mathrm{D_i} = \mathrm{R_i} + 5 \times random(TWK, 3 \times TWK)$$

where R_i is the release date. By assigning due dates according to the random number generated from total work-content, the original strong connect between the release date and the due date in Wilbrecht and Prescott's work can be eliminated. This simulation has 50 replications and each replication has 50000 Monte Carlo trials. The measurement of the performances for the selected 10 production objectives is summarized in Table 6.3.

Table 6.3 Performance measures used in the experiment

Performance Name	Performance Measurement
Fairness	$\sum_{i=1}^{n} \frac{\lvert jobReleaseOrderIndex_i - jobStartOrderIndex_i \rvert}{n}$
Priority Conformity	$\sum_{i=1}^{n} \frac{\lvert jobPriority_i - jobStartOrderIndex_i \rvert}{n}$
Job Lateness Variance	$\sum_{i=1}^{n} \frac{(jobDueDate_i - jobCompletionTime_i - \sum_{i=1}^{n} \frac{jobDueDate_i - jobCompletionTime_i}{n})^2}{n}$
Makespan	$completionTimeOfLastFinishedJob - startTimeOfFirstReleasedJob$
Maximum Tardiness	$Max \begin{Bmatrix} Max\{0, \lvert jobCompletionTime_1 - jobDueDate_1 \rvert\} \\ Max\{0, \lvert jobCompletionTime_2 - jobDueDate_2 \rvert\} \\ \dots \\ Max\{0, \lvert jobCompletionTime_m - jobDueDate_m \rvert\} \end{Bmatrix}$
Mean Respond Time	$\sum_{i=1}^{n} \frac{jobStartTime_i - jobReleaseDate_i}{n}$
Mean Setup Time	$\sum_{i=1}^{n} \frac{jobSetupTime_i}{n}$
Waiting Time Variance	$\sum_{i=1}^{n} \frac{(jobWaitingTime_i - \sum_{i=1}^{n} \frac{jobWaitingTime_i}{n})^2}{n}$

The performance measurements are all designed to have a lower value for better performances so that the performance measurements can be easily combined. Additionally, all the performance measurement values are rounded to between 0 and 1. This is because when the combined objectives are used, the performance measures from different objectives should have the same scale. The explanation of each performance measure is narrated below.

- Fairness: defined for identifying the FCFS rule. It measures the average differences

between the release order number and the start order number of jobs.

- Priority conformity: defined for identifying the W (weight) rule. It measures the average differences between the jobs' priority value (or weight) and their start order number.
- Job lateness variance: defined for identifying the ODD rule. It measures the variance of the jobs' lateness.
- Makespan: defined for identifying the MWKR rule and the MOPNR rule. It measures the difference between the start time of the first release job and the completion time of the last finished job.
- Maximum Tardiness: defined for identifying the EDD rule and the ODD rule. It measures maximum tardiness of all jobs.
- Mean respond time: defined for identifying the LCFS rule. It measures the mean difference between the jobs' start time and release time.
- Mean setup time: defined for identifying the SST rule. It measures the mean setup time for all the jobs.
- Waiting time variance: defined for identifying the LWT rule. It measures the waiting time variance among all jobs.

6.6.3 Sensitivity Analysis to Configure the Threshold

To configure the threshold values for controlling the tree traversal algorithm, sensitivity analysis has been carried out to test the semantic similarities for different threshold values. The sensitivity analysis tested all the production objectives against each one of the dispatching rules.

The combinations of two individual objectives and the combinations of three individual objectives are also tested. The threshold values tested are selected between 0.1 and 1.1. According to the simulation-based dispatching rule selection results, the threshold can be configured so that the better rules can be selected. Table 6.4 shows the results for dispatching rule selection with one production objective with the simulation results. For simplicity, Table 6.5 only shows the semantics-based dispatching rule selection results. In Table 6.4, the first line for each production objective represents the simulation results (i.e. performances listed in Table 6.3) and the second line shows the semantic similarity values and the selection of dispatching rules (shown in bold). From the simulation results captured in the first lines of each objective, the best dispatching rule can be identified. Based on the simulation results, the threshold values can be defined.

For example, a sensitivity analysis for the combination of the "Maximize Fairness" objective and the "Minimize Makespan" objective is shown in Figure 6.12. For this combined objective, the FCFS, the MWKR, and the MOPNR rules have relatively higher semantic similarity values. According to the simulation results with one objective shown in Table 6.4, the FCFS rule has good performance for the "Maximize Fairness" objective. The MWKR rule and the MOPNR rule have good performances for the "Minimize Makespan" objective. Therefore, the threshold should be defined so that only these three rules can be selected. In this case, the threshold value selected is 0.7. The same process is repeated for all other combinations of the objectives.

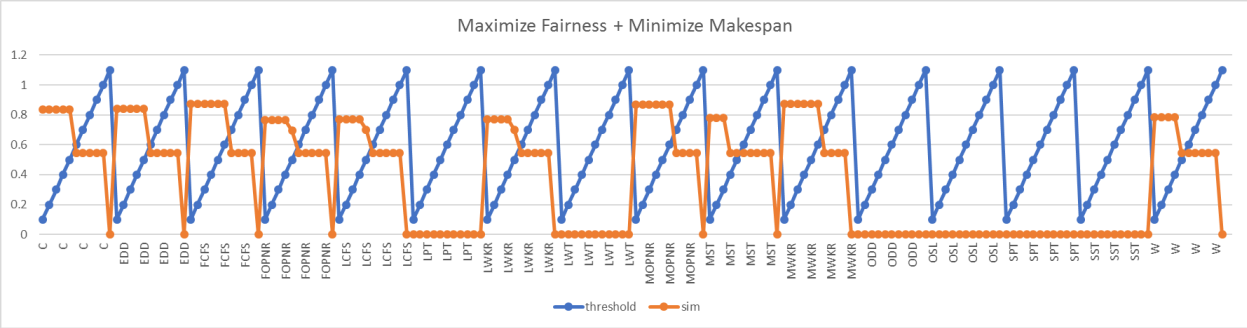


Figure 6.12 Sensitivity analysis to configure the thresholds

6.6.4 Results

As mentioned, the dispatching rule selection results for one objective are shown in Table 6.4. The performances of the combination of dispatching rules are also tested. Using the combined "Maximize Fairness" and the "Minimize Makespan" objective as an example, Figure 6.13 shows the performances of each individual dispatching rules and the combined rule. Based on the semantic similarity values listed in Table 6.5, the combination of the dispatching rule is

$$f = 0.334rank(FCFS) + 0.332rank(MOPNR) + 0.334rank(MWKR)$$

It can be observed from the simulation results in Figure 6.13, the combination of the dispatching rule has better performance than the individual rules.

Table 6.4 Comparison between the results from the proposed approach and the ones from simulation

	1/C	EDD	FCFS	FOPNR	LCFS	LPT	LWKR	LWT	MOPNR	MST	MWKR	ODD	OSL	SPT	SST	W
Max Fairness	110.5	82.5	42.7	110.4	68.1	113.9	107.2	51.5	104.4	84.0	107.7	77.4	92.5	88.8	84.4	110.5
	0.545	0.545	**1.0**	0.545	0.545	0	0.545	0	0.545	0.545	0.545	0	0	0	0	0.545
Max Priority Confm	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	20.8	6.1
	0.545	0.545	0.545	0.545	0.545	0	0.545	0	0	0.545	0.545	0	0	0	0	**1.0**
Min Lateness Var	104.8	27.6	94.4	77.6	99.2	108.4	72.7	72.5	111.8	27.8	118.1	26.6	29.4	83.3	92.6	104.7
	0	0	0	0	0	0.545	0	0.545	0	0	0	**1.0**	0.545	0.545	0.545	0
Min Makespan	280.8	291.4	281.7	302.4	274.6	286.5	302.3	269.3	260.4	287.7	260.7	288.0	294.6	276.3	269.5	280.8
	0.545	0.545	0.545	0.545	0.545	0	0.545	0	**1.0**	0.545	**1.0**	0	0	0	0	0.545
Min Tardiness	302.5	202.1	302.4	275.4	284.8	294.9	266.5	257.1	273.7	201.2	278.2	200.6	206.6	274.1	291.0	302.5
	0.545	**1.0**	0.545	0.545	0.545	0.545	0.545	0.545	0.545	0.545	0.545	**1.0**	0.545	0.545	0.545	0.545
Min Resp. Time	34.2	26.4	42.2	32.5	12.3	33.2	31.1	20.7	31.1	22.1	33.2	25.5	27.8	24.5	27.1	34.2
	0.545	0.545	0.545	0.545	**1.0**	0	0.545	0	0.545	0.545	0	0	0	0	0	0.545
Min Stp. Time	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.83	2.04	2.83
	0	0	0	0	0	0.545	0	0.545	0	0	0	0.545	0.545	0.545	**1.0**	0
Min Waitingtime	191.2	103.5	149.9	103.7	211.1	234.9	98.7	92.2	348.2	112.4	370.7	106.3	123.4	165.7	145.5	191.4
Var	0	0	0	0	0	0.545	0	**1.0**	0	0	0	0.545	0.545	0.545	0.545	0

Table 6.5 Semantics-based dispatching rule selection results with two production objectives

	1/C	EDD	FCFS	FOPNR	LCFS	LPT	LWKR	LWT	MOPNR	MST	MWKR	ODD	OSL	SPT	SST	W
Max Fairness + Max Pri. Conformity	0.545	0.545	**0.868**	0.545	0.545	0	0.545	0	0.545	0.545	0.545	0	0	0	0	**0.868**
Min Lateness Var + Min Mn. Setup Time	0	0	0	0	0	0.545	0	0.545	0	0	0	**0.855**	0.545	0.545	**0.855**	0
Max Pri. Conf. + Min Mn. Resp. Time	0.545	0.545	0.545	0.545	**0.849**	0	0.545	0	0.545	0.545	0.545	0	0	0	0	**0.849**
Min Makespan + Max Fairness	0.545	0.545	**0.873**	0.545	0.545	0	0.545	0	**0.868**	0.545	**0.873**	0	0	0	0	0.545
Min Max Tardiness + Min Mn. Setup Time	0	0	0	0	0	0.545	0	0.545	0	0.545	0	**0.855**	0.545	0.545	**0.855**	0

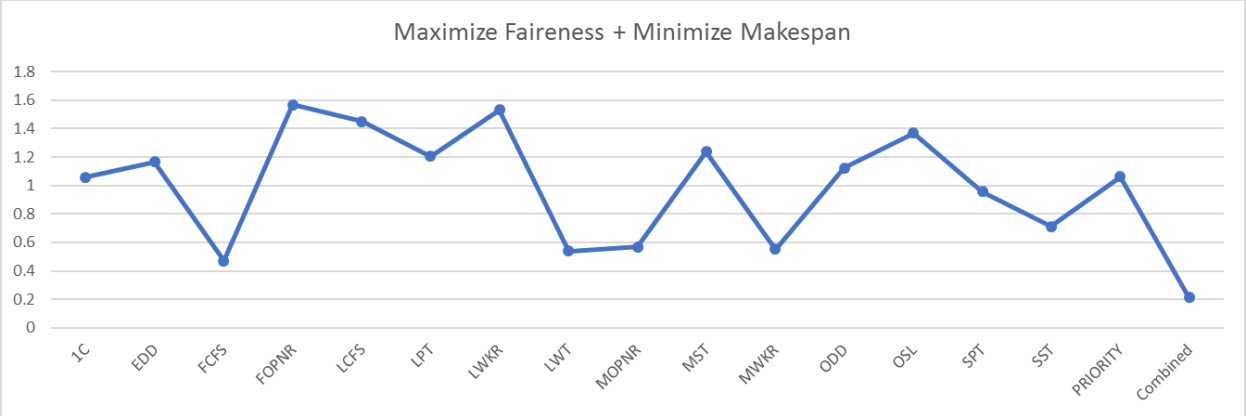


Figure 6.13 Simulation results for the combination of the "Maximize Fairness" and the "Minimize Makespan" objectives

CHAPTER 7. Conclusion

In this chapter, the conclusion of this dissertation is presented. The research work presented in this dissertation is first summarized. Then the research contributions of this dissertation are described. Finally, the limitation of the research work is discussed.

7.1 Summary

Smart Manufacturing (SM) has become a new production paradigm that is pursued by both industry and academia. It requires to develop standardized models to enable the accessibility and availability of computational models to a wide range of industrial users. It also requires computational models to be smoothly integrated with enterprise-wide data and to be properly incorporated human knowledge for efficient decision-making.

To achieve this, it is crucial to develop a method to support the lifecycle activities of computational models like model development, deployment, and retrieval. However, the current standardized computational models can only capture the computational models, and they do not capture the corresponding domain knowledge that can support their lifecycle activities. The lack of interoperability and traceability of domain knowledge in standardized computational models greatly limits the lifecycle activities of computational models by industrial users. The major reason is that the lack of formally represented knowledge in standardized computational models makes the development, deployment, and retrieval of computational models difficult for software tools to carry out automatically and it leaves these lifecycle activities to manual work.

This dissertation proposes a Knowledge Enriched Computational Model (KECM) to

formally capture domain knowledge and integrate that knowledge with standardized computational models to support lifecycle activities of computational models. In this model, the domain knowledge is captured into information model(s), physics-based model(s) and rationales/rules. The information model(s) can be used to explicitly express the domain meaning of a computational model's entities. The physics-based model(s) can capture the physics or behavioral information of an SM system. The rationales/rules can be used to describe the rationality of a computational model and to guide the lifecycle activities of computational models. Semantic links are used to connect these models to the standardized computational model. To implement the KECM, text-based information interchange languages like XML, JSON, and OWL can be used.

To support the development of computational models in distributed environments, the KECM is used as a medium to support formal communication between model developers. Each model developer can update a computational model and add the corresponding knowledge. A case study scenario, which developed a Bayesian Network (BN) model, has been used to validate the proposed method. A KECM model has been developed to support the information exchange between domain experts and data analysts. Due to the BN model and the knowledge used to develop the BN being formally represented, automation of the BN construction has been enabled. The BN can also be extracted by developed parsers for further learning and testing. The utilization of the proposed KECM has reduced the cycle time for the development of the BN and it can eliminate human errors.

This dissertation has discussed two perspectives of model deployment. The first perspective

discusses the data integration between a computational model and an SM system using the KECM. Through the KECM, data in manufacturing systems can be smoothly integrated with the input/output data of a computational model. A case study was developed to deploy a Constraint Programming scheduling optimization model in a B2MML-based system. Through the KECM, input data like job and machine information can be successfully loaded to the standardized computational model; and the output schedule can be also loaded back to the B2MML-based system. The second perspective of model deployment that this dissertation discusses is the combination of multiple models. Three basic types of model combination have been discussed: sequential models, parallel models, and composed models. A general method to formally represent model combinations has been presented. A case study has been developed to demonstrate the composition of an Agent-based model and a Decision Tree model for real-time scheduling. In the case study, the Decision Tree selects the dispatching rule according to the system status. Different data types for representing dispatching rules of the two models have been connected through modeling rules. The consumption of the KECM has been partially implemented, and it proves that the proposed KECM can be easily consumed by software tools.

To support model retrieval, this dissertation proposes a semantics-based method. By formally describing the computational models and the model retrieval requirements in formal semantic expressions, a semantic similarity-based method can be enabled to measure the similarity between them. If the similarity value can satisfy the threshold as defined, the computational model can be retrieved. To support this method, a semantics-based dispatching rule model selection approach has been presented. The formal semantic expressions of

dispatching rules and production objectives have been developed. A tree-based semantic similarity measure has been proposed to calculate the similarity between the given production objective(s) and each single dispatching rule. A combination of dispatching rule method has also been introduced to combine the retrieved individual dispatching rules. This semantics-based dispatching rule model retrieval method has been validated with simulation-based experiments and sensitivity analysis.

7.2 Research Contribution

This dissertation proposes a Knowledge Enriched Computational Model (KECM) to capture and integrate domain knowledge with standardized computational models to support the lifecycle activities of computational models. It fills the research gap of a lack of formally represented domain knowledge integrated with standardized computational models. KECMs have been developed to support several lifecycle applications of computational models. In these applications, the KECM demonstrates the capability to support the development, deployment, and retrieval of computational models. The contributions of these individual applications are:

- Implementation of a KECM to support the development of a Bayesian Network model;
- Implementation of a KECM to support the data integration between an optimization model and a B2MML-based manufacturing system;
- Implementation of a KECM to support the model combination between an Agent-based model and a Decision Tree model;
- Development of a dispatching rule model retrieval method for job shop scheduling using

the KECM.

7.3 Discussion and Limitation

This dissertation proposes a Knowledge Enriched Computational Model to support lifecycle activities of computational models. This dissertation selects OWL as the implementation to implement the KECM. Even though OWL has richer semantics than XML and JSON, OWL also has limitations. OWL does not provide primitive support for managing processes (e.g., workflows) and collection types (e.g., arrays, lists, and hash tables, etc.). It is true that they can be defined in OWL by users, but the reasoning about them is limited. This calls for more plug-ins or official releases to be developed for richer semantics.

This work greatly relies on standardized models. However, currently, not all computational models have their own standardized models in text-based model interchange formats. This means that there is no uniform way to enable the interoperability of these models. But to achieve the Smart Manufacturing's goal to have accessibility and availability of computational models, the standardized computational models must be developed. To allow plug-and-play capability, the interoperability of a type of computational model among only two or three specific software tools is not acceptable. This calls for the development of more standardized computational models.

APPENDIX – A

To facilitate the production of all 26 letters on the production line (as mentioned in section **Error! Reference source not found.**), the production sequences of all 26 letters are listed in REF _Ref517010411 \h * MERGEFORMAT Table 0.1. Trentesaux et al. (2013) presented the production sequences of 7 letters. This dissertation expands their production sequences of 7 letters to cover all 26 letters.

Table 0.1 Letter part production sequence

Letter	A	B	C	D	E	F	G	H	I
Sequence	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp
	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp
	Axis_comp	Axis_comp	L_comp	Axis_comp	Axis_comp	L_comp	Axis_comp	Axis_comp	I_comp
	r_comp	r_comp	L_comp	L_comp	r_comp	r_comp	r_comp	Axis_comp	Screw_comp
	L_comp	r_comp		L_comp	r_comp		L_comp	r_comp	
	I_comp	I_comp		I_comp	L_comp			I_comp	
	Screw_comp	Screw_comp		Screw_comp				I_comp	
								Screw_comp	
								Screw_comp	

Letter	J	K	L	M	N	O	P	Q	R
Sequence	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp
	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp
	L_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	L_comp	r_comp	Axis_comp	Axis_comp
		r_comp	I_comp	Axis_comp	L_comp	L_comp	L_comp	r_comp	r_comp
		r_comp	I_comp	r_comp	I_comp			L_comp	r_comp
		I_comp	Screw_comp	L_comp	Screw_comp			L_comp	I_comp
		Screw_comp	Screw_comp	I_comp					Screw_comp
				Screw_comp					

Letter	S	T	U	V	W	X	Y	Z	
Sequence	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	
	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	Axis_comp	
	Axis_comp	r_comp	Axis_comp	Axis_comp	Axis_comp	L_comp	Axis_comp	Axis_comp	
	Axis_comp	L_comp	L_comp	r_comp	Axis_comp	L_comp	r_comp	Axis_comp	
	r_comp		I_comp	r_comp	r_comp	L_comp	L_comp	r_comp	
	r_comp		Screw_comp	r_comp	L_comp	L_comp		r_comp	
	r_comp			r_comp	I_comp			r_comp	
					Screw_comp				

REFERENCES

ANSI/ISA (2010). *ANSI/ISA-95.00.01-2010 (IEC 62264-1 Mod) Enterprise-Control System Integration - Part 1: Models and Terminology*. USA: ANSI/ISA.

Assouroko, I., & Denno, P. (2016). A metamodel for optimization problems. NIST Interagency/Internal Report (NISTIR) – 8096.

Azadeh, A., Hosseini, N., Abdolhossein Zadeh, S., & Jalalvand, F. (2015). A hybrid computer simulation-adaptive neuro-fuzzy inference system algorithm for optimization of dispatching rule selection in job shop scheduling problems under uncertainty. *The International Journal of Advanced Manufacturing Technology, 79*(1), 135–145.

Azadeh, A., Negahban, A., & Moghaddam, M. (2012). A hybrid computer simulation-artificial neural network algorithm for optimization of dispatching rule selection in stochastic job shop scheduling problems. *International Journal of Production Research, 50*(2), 551–566.

Baykasoglu, A., Göçken, M., & Özbakir, L. (2010). Genetic programming based data mining approach to dispatching rule selection in a simulated job shop. *Simulation, 86*(12), 715–728.

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE*. Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd.

Bellifemine, F., Bergenti, F., Caire, G., & Poggi A. (2005). Jade - A Java Agent Development

Framework. In: Bordini R.H., Dastani M., Dix J., El Fallah Seghrouchni A. (eds), *Multi-Agent Programming. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series)* (vol 15). Boston: Springer.

Blackstone, J. H., Phillips, D. T., & Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research, 20*(1), 27–45.

Boothroyd, G., Dewhurst, P., & Knight, W. A. (2011). *Product Design for Manufacture and Assembly* (3rd ed.). Boca Raton, FL: CRC Press.

Brodsky, A., Shao, G., & Riddick, F. (2016). Process analytics formalism for decision guidance in sustainable manufacturing. *Journal of Intelligent Manufacturing, 27*, 561-580.

Campos, L. M., & Castellano, J. G. (2007). Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning, 45*(2), 233-254.

Chapman, S. N. (2006). *Fundamentals of production planning and control*. Upper Saddle River, New Jersey, 07458: Pearson Education, Inc.

Chen, X., Wen, L. H., & Murata, T. (2012). Composite dispatching rule design for dynamic scheduling with customer-oriented production priority control. *IEEJ Transactions on Electrical and Electronic Engineering, 7*(1), 53–61.

Chungoora, N., Young, R. I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N. A., Cutting-Decelle, A., Harding, J. A., & Case, K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry, 64*(4), 392-401.

Davis, J., Edgar, T., Graybill, R., Korambath, P., Schott, B., Swink, D., Wang, J., & Wetzel, J. (2015). Smart manufacturing. *Annual Review of Chemical and Biomolecular Engineering, 6*, 141-160.

Denno, P., & Kim, D. B. (2016). Integrating views of properties in models of unit manufacturing processes. *International Journal of Computer Integrated Manufacturing, 29*(9), 996-1006.

DMG (2015, November 10). Portable Format for Analytics (PFA). Retrieved from http://dmg.org/pfa/index.html.

DMG (2016, August). *PMML Version 4.3*. Retrieved from http://dmg.org/pmml/pmml-v4-3.html.

ECSS (2011, January 25). *Simulation modelling platform – Volume 1: Principles and requirements*. Retrieved from http://www.ecss.nl/wp-content/uploads/standards/ecss-e/ECSS-E-TM-40-07-Volume1A25January2011.pdf.

El-Bouri, A., & Amin, G.R. (2015). A combined OWA-DEA method for dispatching rule selection. *Computers & Industrial Engineering, 88*, 470–478.

El-Bouri, A., & Shah, P. (2006). A neural network for dispatching rule selection in a job shop. *The International Journal of Advanced Manufacturing Technology, 31*(3), 342–349.

FIPA (2002). *FIPA Contract Net Interaction Protocol Specification*. Geneva, Switzerland: FIPA.

Geiger, C. D., Uzsoy, R., & Aytuğ, H. (2006). Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling, 9*(1), 7–34.

Harispe, S., Ranwez, S., Janaqi, S., & Montmain, J. (2017). Semantic similarity from natural

language and ontology analysis. *arXiv:1704.05295*.

Hartmann, T., Moawad, A., Fouquet, F., Nain, G., Klein, J., Traon, Y. L., & Jezequel, J. M. (2017). Model-driven analytics: connecting data, domain knowledge, and learning. arXiv:1704.01320.

Haupt, R. (1989). A survey of priority rule-based scheduling. *Operations-Research-Spektrum, 11*(1), 3–16.

Heger, J., Hildebrandt, T., & Scholz-Reiter, B. (2015). Dispatching rule selection with Gaussian processes. *Central European Journal of Operations Research, 23*(1), 235–249.

Henkel, R., Endler, L., Peters, A., Le Novère, N., & Waltemath, D. (2010). Ranked retrieval of Computational Biology models. *BMC Bioinformatics, 11:*423.

Hoehndorf, R., Dumontier, M., Gennari, J. H., Wimalaratne, S., de Bono, B., Cook, D. L., & Gkoutos, G. (2011). Integrating systems biology models and biomedical ontologies. *BMC Systems Biology, 5:*124.

Horridge, M., & Bechhofer, S. (2011). The OWL API: A Java API for OWL ontologies. *Semantic Web, 2*(1), 11-21.

Johnson, I., Abécassis, J., Charnomordic, B., Destercke, S., & Thomopoulos, R. (2010). Making ontology-based knowledge and decision trees interact: An approach to enrich knowledge and increase expert confidence in data-driven models. In: Bi Y., Williams MA. (eds), *Knowledge Science, Engineering and Management. KSEM 2010. Lecture Notes in Computer Science* (vol. 6291). Berlin: Springer.

Joseph, O. A., & Sridharan, R. (2011). Effects of routing flexibility, sequencing flexibility and

scheduling decision rules on the performance of a flexible manufacturing system. *The International Journal of Advanced Manufacturing Technology, 56*(1), 291–306.

Kalet, A. M., Doctor, J. N., Gennari, J. H., & Phillips, M. H. (2017). Developing Bayesian networks from a dependency-layered ontology: A proof-of-concept in radiation oncology. *Medical Physics, 44*(8), 4350-4359.

Kannan, K., Srivastava, B., -Sosa, R. U., Schloss, R. J., & Liu, X. (2014). SemEnAl: using semantics for accelerating environmental analytical model discovery. In: Srinivasa S., Mehta S. (eds), *Big Data Analytics. BDA 2014. Lecture Notes in Computer Science* (vol 8883). Cham: Springer.

Kashfi, M. A., & Javadi, M. (2015). A model for selecting suitable dispatching rule in FMS based on fuzzy multi attribute group decision making. *Production Engineering, 9*(2), 237–246.

Kızıl, M., Özbayrak, M., & Papadopoulou, T. C. (2006). Evaluation of dispatching rules for cellular manufacturing. *The International Journal of Advanced Manufacturing Technology, 28*(9), 985–992.

Kulkarni, A., Balasubramanian, D., Karsai, G., Narayanan, A., & Denno, P. (2016). A domain-specific language for model composition and verification of multidisciplinary models. Paper presented at 2016 Conference on Systems Engineering Research, Huntsville, AL, USA.

Le Novère, N., Finney, A., Hucka. M., Bhalla. U. S., Campagne F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J. L.,

Spence, H. D., & Wanner, B. L. (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology, 23*(12), 1509-1515.

Lechevalier, D., Narayanan, A., Rachuri, S., Foufou, S., & Lee Y. T. (2016). Model-based engineering for the integration of manufacturing systems with advanced analytics. In: Harik R., Rivest L., Bernard A., Eynard B., Bouras A. (eds), *Product Lifecycle Management for Digital Transformation of Industries. PLM 2016. IFIP Advances in Information and Communication Technology* (vol. 492). Cham: Springer.

Lemaignan, S., Siadat, A., Dantan, A. -Y., & Semenenko, A. (2006). Mason: a proposal for an ontology of manufacturing domain. Paper presented at IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06), Prague, Czech Republic. doi: 10.1109/DIS.2006.48.

Li, Y., Thomas, M. A., & Osei-Bryson, K. (2017). Ontology-based data mining model management for self-service knowledge discovery. *Information Systems Frontiers, 19*(4), 925-943.

Li, Y., Zhang, H., Roy, U., & Lee, Y. T. (2017). A data-driven approach for improving sustainability assessment in advanced manufacturing. Paper presented at 2017 IEEE International Conference on Big Data (BigData 2017), Boston, MA, USA.

Lin, Y., Chiu, C., & Tsai, C. (2008). The study of applying ANP model to assess dispatching rules for wafer fabrication. *Expert Systems with Applications, 34*(3), 2148–2163.

Liu, H., & Dong, J. (1996). Dispatching rule selection using artificial neural networks for dynamic planning and scheduling. *Journal of Intelligent Manufacturing, 7*, 243–250.

Madan, J., Mani, M., & Lyons, K. W. (2013). Characterizing energy consumption of the injection molding process. Paper presented at Proceedings from ASME 2013: Manufacturing Science and Engineering Conference, Madison, WI, USA. doi: 10.1115/MSEC2013-1222.

MESA International (2013). B2MML V0600. Retrieved from https://services.mesa.org/ResourceLibrary/ShowResource/0f47758b-60f0-40c6-a71b-fa7b2363fb3a.

Mouelhi-Chibani, W., & Pierreval, H. (2010). Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering, 58*(2), 249–256.

Munger, T., Desa, S., & Wong, C. (2015). The use of domain knowledge models for effective data mining of unstructured customer service data in engineering applications. Paper presented at 2015 IEEE First International Conference on Big Data Computing Service and Applications, Redwood City, CA, USA. doi: 10.1109/BigDataService.2015.46.

Muñoz, E., Capón-García, E., Laínez-Aguirre, J. M., Espuña, A., & Puigjaner, L. (2014). Using mathematical knowledge management to support integrated decision-making in the enterprise. *Computers and Chemical Engineering, 66*, 139-150.

Muñoz, E., Capón-García, E., Laínez-Aguirre, J. M., Espuña, A., & Puigjaner, L. (2012). Mathematical knowledge management for enterprise decision making, *Computer Aided Chemical Engineering, 32*, 637-642.

Muñoz, E., Capón-García, E., Laínez-Aguirre, J. M., Espuña, A., & Puigjaner, L. (2013). Integration of enterprise levels based on an ontological framework. *Chemical*

Engineering Research and Design, 91, 1542-1556.

Nannapaneni, S., Mahadevan, S., & Rachuri, S. (2016). Performance evaluation of a manufacturing process under uncertainty using Bayesian networks. *Journal of Cleaner Production, 113*(1), 947-959.

Oberg, E., Jones, F. D., Horton, H. L., Ryffel, H. H., & McCauley, C. J. (2004). *Machinery's handbook* (27th ed.). New York, NY: Industrial Press, Inc.

Özgüven, C., Özbakır, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling, 34*, 1539-1548.

Panwalkar, S. S., & Iskander, W. (1977). A survey of scheduling rules. *Operations Research, 25*(1), 45–61.

Perez-Rey, D., Anguita, A., & Crespo, J. (2006). OntoDataClean: Ontology-based integration and preprocessing of distributed data. In: Maglaveras N., Chouvarda I., Koutkias V., Brause R. (eds), *Biological and Medical Data Analysis. ISBMDA 2006. Lecture Notes in Computer Science* (vol. 4345). Berlin Springer.

Pinedo, M. L. (2010). *Scheduling theory, algorithms, and systems* (4th ed.). 223 Spring Street, New York, NY 10013, USA: Springer Science+Business Media, LLC.

Pivarski, J., Bennett, C., & Grossman, R. L. (2016). Deploying Analytics with the Portable Format for Analytics (PFA). Paper presented at Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA. doi: 10.1145/2939672.2939731.

Pras, A., & Schoenwaelder, J. (2003). On the difference between information models and data models. *RFC 3444, Jan. 2003*.

Ribeiro, I., Peças, P., & Henriques, E. (2012). Assessment of energy consumption in injection moulding process. In: Dornfeld, D., and Linke B. (eds.), *Leveraging Technology for a Sustainable World*. Berlin, Springer.

Scholz-Reiter, B., Heger, J., & Hildebrandt, T. (2010). Gaussian processes for dispatching rule selection in production scheduling: Comparison of learning techniques. Paper presented at 2010 IEEE International Conference on Data Mining Workshops, Sydney, NSW, Australia. https://doi.org/10.1109/ICDMW.2010.19.

Schulz, M., Krause, F., Le Novère, N., Klipp, E., & Liebermeister, W. (2011). Retrieval, alignment, and clustering of computational models based on semantic annotations. *Molecular Systems Biology, 7:*512.

Scutari, M., & Denis, J. B. (2014). *Bayesian networks with examples in R*. Boca Raton, FL: Taylor & Francis Group.

Shafiq, S. I., Faheem, M., & Ali, M. (2010). Effect of scheduling and manufacturing flexibility on the performance of FMS. *Global Journal of Flexible Systems Management, 11*(3), 21–38.

Shao G., Denno, P., Jones, A., & Lu, Y. (2016). Implementing the ISO 15746 standard for chemical process optimization. Paper presented at ASME 2016 11th International Manufacturing Science and Engineering Conference, Blacksburg, Virginia, USA. doi: 10.1115/MSEC2016-8635.

Shiue, Y. (2009). Data-mining-based dynamic dispatching rule selection mechanism for shop floor control systems using a support vector machine approach. *International Journal of Production Research, 47*(13), 3669–3690.

Sinha, A.P., & Zhao, H. (2008). Incorporating domain knowledge into data mining classifiers: An application in indirect lending. *Decision Support Systems, 46*(1), 287-299.

SMLC (2011). Implementing 21 Century Smart Manufacturing, workshop summary report. Retrieved from https://smartmanufacturingcoalition.org/sites/default/files/implementing_21st_century_smart_manufacturing_report_2011_0.pdf.

Stanford Center for Biomedical Informatics Research (BMIR). (2018). Protégé: A free, open-source ontology editor and framework for building intelligent systems. http://protege.stanford.edu/. Accessed on 29 Jan 2018.

Suresh, P., Joglekar, G., Hsu, S., Akkisetty, P., Hailemariam, L., Jain, A., Reklaitis, G., & Venkatasubramanian, V. (2008). Onto MODEL: Ontological mathematical modeling knowledge management. *Computer Aided Chemical Engineering, 25*, 985-990.

Szabo, C., & Teo, Y. M. (2011). An approach to semantic-based model discovery and selection. Paper presented at Proceedings of the 2011 Winter Simulation Conference, Phoenix, Arizona, USA.

The Apache Software Foundation. (2018). Apache jena: A free and open source java framework for building semantic web and linked data application. https://jena.apache.org/. Accessed on 29 Jan 2018.

Trentesaux, D., Pach, C., Bekrar, A., Sallez, Y., Berger, T., Bonte, T., Leitão, P., & Barbosa, J.

(2013). Benchmarking flexible job-shop scheduling and control systems. *Control Engineering Practice, 21*, 1204-1225.

Usman, Z. (2013). Towards a formal manufacturing reference ontology. *International Journal of Production Research, 51*(22), 6553-6572.

Veryard, R. (1992). *Information modelling: practical guidance*. New York: Prentice Hall.

W3C (2000, Nov. 21). *XEXPR - A Scripting Language for XML*. Retrieved from https://www.w3.org/TR/2000/NOTE-xexpr-20001121/.

W3C (2004, April 10). *Mathematical Markup Language (MathML) Version 3.0 2nd Edition*. Retrieved from https://www.w3.org/TR/MathML3/.

W3C (2004, May 21). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. Retrieved from https://www.w3.org/Submission/SWRL/.

W3C (2012, Dec. 11). OWL 2 Web Ontology Language Primer (Second Edition). Retrieved from https://www.w3.org/TR/owl2-primer/.

W3C Working Group. (2012). OWL 2 Web Ontology language Manchester syntax (2nd ed.). W3C Recommendation.

Wadhams, J. (2015, Oct. 20). *JsonLogic - Build complex rules, serialize them as JSON, share them between front-end and back-end*. Retrieved from http://jsonlogic.com/.

Wilbrecht, J. K., & Prescott, W. B. (1969). The influence of setup time on job shop performance. *Management Science, 16*(4), 274–280.

Witherell, P., Krishnamurty, S., & Grosse, I. R. (2007). Ontologies for supporting engineering design optimization. *Journal of Computing and Information Science in Engineering, 7*,

141-150.

Wu, Z., & Palmer, M. (1994). Verb semantics and lexical selection. Paper presented at Proceeding ACL ’94 Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, Las Cruces, New Mexico, USA.

Zhang, H., & Roy, U. (2018). A semantics-based dispatching rule selection approach for job shop scheduling. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-018-1421-z.

Zhang, H., Zhu, B., Li, Y., Yaman, O., & Roy, U. (2015). Development and utilization of a Process-oriented Information Model for sustainable manufacturing. *Journal of Manufacturing Systems, 37*(2), 459-466.

Zhong, R. Y., Huang, G. Q., Dai, Q. Y., & Zhang, T. (2014). Mining SOTs and dispatching rules from RFID-enabled real-time shopfloor production data. *Journal of Intelligent Manufacturing, 25*, 825–843.

VITA

HENG ZHANG

211 Lafayette Rd Apt 620
Syracuse, NY 13205

Email: hzhang33@syr.edu
Phone: 315-289-8288

EDUCATION

Doctor of Philosophy in Mechanical Engineering 01/2013 – 08/2018
College of Engineering & Computer Science, Syracuse University

Master of Science in Mechanical Engineering 08/2011 – 12/2012
College of Engineering & Computer Science, Syracuse University

Bachelor of Engineering in Mechanical Engineering 09/2006 – 06/2010
School of Mechanical Engineering, Hebei University of Technology

WORK EXPERIENCE

Research Assistant at College of Engineering & Computer Science, Syracuse University
01/2018 – Present

Teaching Assistant at College of Engineering & Computer Science, Syracuse University
09/2013 – 05/2017

Research Assistant at College of Engineering & Computer Science, Syracuse University
01/2013 – 05/2013

ACADEMIC PUBLICATIONS

1. **Zhang, H.**, & Roy, U. (2018). A semantics-based dispatching rule selection approach for job shop scheduling. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-018-1421-z.
2. Li, Y., **Zhang, H.**, Roy, U., & Lee, Y.T. (2017). A Data-Driven Approach for Improving Sustainability Assessment in Advanced Manufacturing. Paper presented at 2017 IEEE International Conference on Big Data, Boston, MA, USA.
3. **Zhang, H.,** & Roy, U. (2017). An Ontology based Information Framework for Smart Manufacturing Systems Interoperability. *C+CA: Progress in Engineering and Science, 42(4)*, 1572-1577.
4. **Zhang, H.**, Zhu, B., Li, Y., Yaman, O., & Roy, U. (2015). Development and Utilization of A Process-oriented Information Model for Sustainable Manufacturing. *Journal of Manufacturing Systems, 37*(2), 459–466.
5. **Zhang, H.,** & Roy, U. (2015). A Semantic Similarity Based Dispatching Rule Selection System for Job Shop Scheduling with Multiple Production Objectives. Proceedings of the ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (ASME), August 2-5, 2015, Boston, Massachusetts, USA.
6. **Zhang, H.,** & Roy, U. (2014). Development of An Information Model for the Integration of Product Design and Sustainability Evaluation. Proceedings of the ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, August 17-20, 2014, Buffalo, New York, USA.

7. Roy, U., Zhu, B., Li, Y., **Zhang, H.,** & Yaman, O. (2014). *Mining Big Data in Manufacturing: Requirement Analysis, Tools and Techniques*. Proceedings of the ASME 2014 International Mechanical Engineering Congress and Exposition, November 14-20, 2014, Montreal, Canada.