Databases and Information Systems IX
G. Arnicans et al. (Eds.)

doi:10.3233/978-1-61499-714-6-57

Ontology-Based Customization of a Scheduling System for Discrete Manufacturing [1]

Jelena SANKO [2], Hele-Mai HAAV and Vahur KOTKAS
Institute of Cybernetics at TUT, Tallinn, Estonia

Abstract. Customization is a very important feature of any manufacturing scheduling system. In many cases large commercial manufacturing scheduling systems are not easily and efficiently customizable to meet requirements of small and medium size enterprises. Therefore, this paper proposes an ontology-based architectural solution for the customization of manufacturing scheduling systems. According to the approach, the input to the scheduling system is a customized manufacturing scheduling ontology that is an extension of the manufacturing scheduling ontology provided in this paper. The customized manufacturing scheduling ontology performs as a knowledge base and data access point for the manufacturing scheduling system and enables it to be easily adapted to different products and their manufacturing processes.

Keywords. ontology, manufacturing scheduling, customization, software development

1. Introduction

Scheduling deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives [1]. The resources, tasks and objectives can take many different forms. The *resources* may be, for example, machines, materials or operators. The *tasks* may be operations (in a plant), take-offs and landings (at an airport), stages (in a construction project) or computer programs. Each *task* has a certain priority level, an earliest possible starting time, a committed shipping due date, a deadline. The *objectives* may be, for example, minimizing the time of completing all tasks or minimizing the number of tasks that are completed after their respective *due dates*.

In this paper, we focus on manufacturing scheduling that deals with allocation of resources to manufacturing tasks that are required by an enterprise to manufacture products that are requested by the customers. In particular, our intention is to design an easily customizable scheduling system that could be exploited in different small and medium size enterprises in order to increase their productivity. The goal of the customization of a manufacturing scheduling system is to provide a system that can be easily adapted and

[1]This research was supported by the Estonian Ministry of Research and Education institutional research grant no. IUT33-13.
[2]Corresponding Author: Jelena Sanko; E-mail: jelena@cs.ioc.ee.

used to scheduling of manufacturing of different products. For that purpose the manufacturing scheduling ontology is designed. It serves as a semantic knowledge base providing a manufacturing scheduling system access to the data and knowledge about the particular product manufacturing process. In this paper, the scope of the domain of manufacturing scheduling ontology is restricted to discrete manufacturing that is an order-oriented manufacturing for product lines, where there is frequent switching from one product to another. In order to be reusable, it is designed to be as general as possible to capture knowledge of discrete manufacturing scheduling domain. There are several high level manufacturing scheduling ontologies available ([2] , [3]). The manufacturing scheduling ontology provided in this paper takes into account some very general concepts of manufacturing ontology described in [2] and extends this set of concepts with many concepts that are specific to the scheduling process of discrete manufacturing. To the best of our knowledge we do not know any such a detailed ontology for capturing knowledge of scheduling of discrete manufacturing.

In order to support customization of the scheduling system we suggest ontology-based solution that uses two ontologies expressed in OWL[3] as follows: the manufacturing scheduling ontology that serves as a general conceptual model of the manufacturing scheduling domain and the customized manufacturing scheduling ontology that uses the latter and provides specific knowledge of the particular manufacturing scheduling process related to a specific product family that share this process. This solution allows the scheduling system to be used for manufacturing of different products by providing the customized manufacturing scheduling ontology as an input to the manufacturing scheduling system.

The main contributions of the paper are the manufacturing scheduling ontology and ontology-driven approach of design and development of manufacturing scheduling software for order-oriented and lean mass customization based manufacturing. The manufacturing scheduling system includes the following novel features: ontology based system architecture and customization of the scheduling system. Software elaboration and evaluation have been done in collaboration with Bolefloor[4] Ltd that produces novel wooden design products (floors, furniture plates etc.) made of boards with naturally curved edges.

We have previous experience with developing an ontology-driven architecture for a manufacturing scheduling system. In [4] we described an informal manufacturing scheduling ontology in order to drive essential system components from the captured ontological knowledge. In this paper, we present a new approach and manufacturing scheduling system architecture that uses the formalized manufacturing scheduling ontology represented in OWL as a main enabler of customization of the manufacturing scheduling system.

The rest of the paper is structured as follows. Section 2 is devoted to the related work and Section 3 provides an overview of the main concepts of manufacturing scheduling ontology used for the discrete manufacturing systems and their relationships. Section 4 gives a general view of an architecture of a scheduling system for discrete manufacturing and its components and describes customization of the scheduling system using the formal manufacturing scheduling ontology. In Section 5, we discuss problems and difficulties involved in developing implementation of the provided architecture. Section 6 concludes the paper.

[3]https://www.w3.org/TR/owl2-primer/

[4]http://www.bolefloor.com/

2. Related Work

During the last years, a large number of different scheduling systems has been developed. Some of scheduling systems are *generic*, which can be applicable to any scheduling problem after some customization, others are *application-specific* systems and research systems (academic prototypes). For example, there is the *Production Planning and Detailed Scheduling System* (PP/DS), which is a part of the *Advanced Planning and Optimization* (APO) software developed by SAP, and is a flexible system that can be adapted easily to many industrial settings. A *Production Scheduler* system developed by i2 Technologies is quite generic and can be adapted to many different manufacturing settings. The *Taylor Scheduling Software* has a number of generic optimization procedures and heuristics built in, including priority rules and local search procedures and provides scheduling solutions to manufacturers worldwide. Other known scheduling software solution providers to refer are *Preactor*, *Orchestrate*, *Global Shop*, *Cybertec*, *AS-PROVA* (an Advanced Planning and Scheduling (APS) system). For detailed information see an overview and examples of scheduling systems in [1]. Typically, introduction, customization and maintenance of these scheduling systems into an operational (small-size) manufacturing need too much effort. We do not know any commercial manufacturing scheduling system that uses ontology-based customization approach. However, there have been some attempts to develop ontologies for scheduling tasks and use these for making scheduling systems more flexible and adaptable.

The OZONE [3] scheduling ontology can be characterised as a meta-model of the domain of scheduling. It provides a language for describing those aspects of the scheduling domain that are relevant to construction of an application system, and a set of constraints on how concepts in the language fit together to form consistent domain models. The OZONE ontology defines five base concepts: *demand*, *activity*, *resource*, *product* and *constraint*. An *activity* is a process that can be executed over a certain time interval and uses resources to produce goods or services required. *Scheduling* is defined as a process of feasibly synchronizing the use of resources by activities to satisfy demand over time. The concept *demand* and *activity* in OZONE have attributes such as *time range* and *assigned-resource*, but they do not specify the number of resources that are required by each demand or activity.

The OZONE scheduling domain ontology is applied for constructing domain models in the COMIREM system [5]. COMIREM is a web-based system devoted to the problem of interactive and dynamic allocation of resources to activities over specific time interval. COMIREM is designed for solving a particular type of scheduling problems, where assigning of complex, heterogeneous sets of resources to the planned activities must be synchronized to satisfy complex constraints. In COMIREM *activities* can be organized hierarchically into multi-level activity networks.

In contrast to the OZONE ontology, Rajpathak et al. [6] propose a *task ontology*, which formally describes scheduling problems, independently of particular application domains and independently of how the problems can be solved. That is, Rajpathak et al. [6] provide a domain-independent and formally specified reference model for scheduling applications. This can be used as the basis for further analysis of the class of scheduling problems and also as a concrete reusable resource to support system development in scheduling applications.

Borgo et al. [2] proposed a *core ontology for the manufacturing domain.* It consists of an ontological classification of ADACOR [7] concepts according to the DOLCE foun-

dational ontology[8]. ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) defines its own proprietary manufacturing ontology, expressed in an object-oriented frame-based manner — that is, uses classes to describe concepts and predicates and fixes them as part of the application ontology.

Lemaignan et al. [9] proposed MASON (Manufacturing Semantics ONtology) ontology that is an upper ontology for manufacturing domain built upon three head concepts: *entities*, *operations* and *resources*.

The provided list of scheduling ontologies in the manufacturing, of course, is not complete.

In contrast to ontologies discussed above, we consider the specific application domain of scheduling — *discrete manufacturing* that is an order-oriented manufacturing for product lines, where there is frequent switching from one product to another. In particular, we consider an experimental lean mass customization based manufacturing system [10], the goal of which is the mass product of unique and personalized products and elimination of the waste from the manufacturing. *Lean manufacturing system* concept was developed primarily in Japan, particularly for the Toyota manufacturing system [11].

3. Ontology for Scheduling of Discrete Manufacturing Domain

A generic manufacturing process can be described by the following scheme. *Customer orders* have to be translated into *operations* with associated *due dates* (committed shipping or completion dates) or *deadlines* (dates, when the *due dates* absolutely must be met). Completion of `Orders` after their *due dates* is allowed, but penalty may be imposed. These *operations* often have to be processed on *machines* (or in a *workcenter*) by *workers* in a given order or sequence. The processing of *operations* may sometimes be *delayed* if individual machines are busy. When high-priority *operations* have to be processed at once, *pre-emptions* may occur. Unexpected events, such as machine breakdowns or longer than expected processing times, may have an essential effect on the schedules [12].

As a starting point for our ontology development, we take the foundational ontology for the manufacturing domain [2] and the OZONE core scheduling ontology [3]. The first gives core concepts on the very high level of abstraction and the second provides a general basis for formulating scheduling domain models. We specify concepts and relationships of the discrete manufacturing scheduling ontology presented in this paper taking into account specific requirements of the discrete manufacturing scheduling domain and the corresponding scheduling software.

We define a *scheduling problem* as a constrained optimization problem, where time-constrained *resources* should be assigned to time-constrained *operations* related to particular *orders*, at a particular time within a predefined *time horizon* of a *schedule* in accordance with predefined *constraints* and *scheduling objectives*. A *time horizon* of a *schedule* defines a time range for which a *schedule* for all *orders* to be scheduled has to be constructed. A *scheduling objective* of a scheduling task is, for example, minimizing the *makespan* (i.e., completion time) of each *order* considered in the scheduling problem or minimizing the number of *orders* that are completed after their respective due dates or to schedule *operations* in such a way as to use available *resources* in an efficient way.

Thus, basic concepts that define ontology of a *manufacturing scheduling* domain are as follows: `Order, Operation, ProcessTemplate, ProcessModel, Resource,`

Schedule and Constraint. By convention, we use Computer Modern Typewriter font to distinguish the specific concepts.

Diagrammatically, we can represent a *manufacturing scheduling ontology* with the following data structure (see Figure 1): the *rectangles* are concepts and *arrows* between them are semantic relationships between these concepts. The arrowheads indicate the direction of the relationships (i.e. their range), the *name of a relationship* is written next to the arrow and *numbers* near the arrowhead represent the *minimum and maximum cardinality* for that relationship. Single number represents exact cardinalities (e.g., "1" for a cardinality of exactly 1), while the asterisk (*) denotes an unrestricted cardinality (e.g., " 1..*" means a minimum cardinality of 1). We assume, that by default the cardinality of the relationship is one or more (1..*) and this cardinality is not shown in a scheme of manufacturing scheduling ontology.

Let us specify basic concepts of the manufacturing scheduling ontology in more detail. For the shake of readability only informal definitions of concepts are given.

Order. Each Order refers to the manufacturing of one particular (type of) *product*. Each Order has ProductData. The ProductData concept may have a set of attributes that characterize an actual product or product family. For example, these attributes can be such as *type* (or *production class*), *material*, *quantity*, *size*. Attributes of the ProductData concept are not given in the manufacturing scheduling ontology because these are dependent on product and are intended to be added when developing a customized manufacturing scheduling ontology.

Resource. In general, Resources specify something or somebody needed to process the Operations (i.e., to manufacture the ordered *product*). At the moment, we take into account only the time-constrained Resources specified by *availability periods* (called Availabilities), when they are available to perform assigned Operations, and by their abilities (called Capabilities) to do a particular type of work, such as *skills* for workers and *functions* for machines.

To ensure that Resources that are already assigned to an Operation will not be assigned to another Operation, these Resources should be subsequently *allocated*. Each Resource has a *status*, that is a changeable attribute indicating the current *status* of the Resource.

Operation. An Operation represents a technological operation performing in a manufacture that requires a certain time for its processing (i.e., has a *duration*).

A *duration* of an Operation depends on a particular Resource or a set of Resources assigned to the Operation, with exception for Operations that do not require any Resources for its processing. Thus, a *duration* is a resource-dependent attribute and an Operation can have different *durations* for different combinations of assigned Resources. The duration of an operation is calculated according to the given duration equation, which result is represented as as an attribute of the ResourceAssignementConstraint concept.

The processing of an Operation (i.e., when an Operation will take place) depends on its predecessor Operations. That is, an Operation can be processed if and only if all its predecessor Operations are finished and it is defined by means of PrecedenceConstraints.

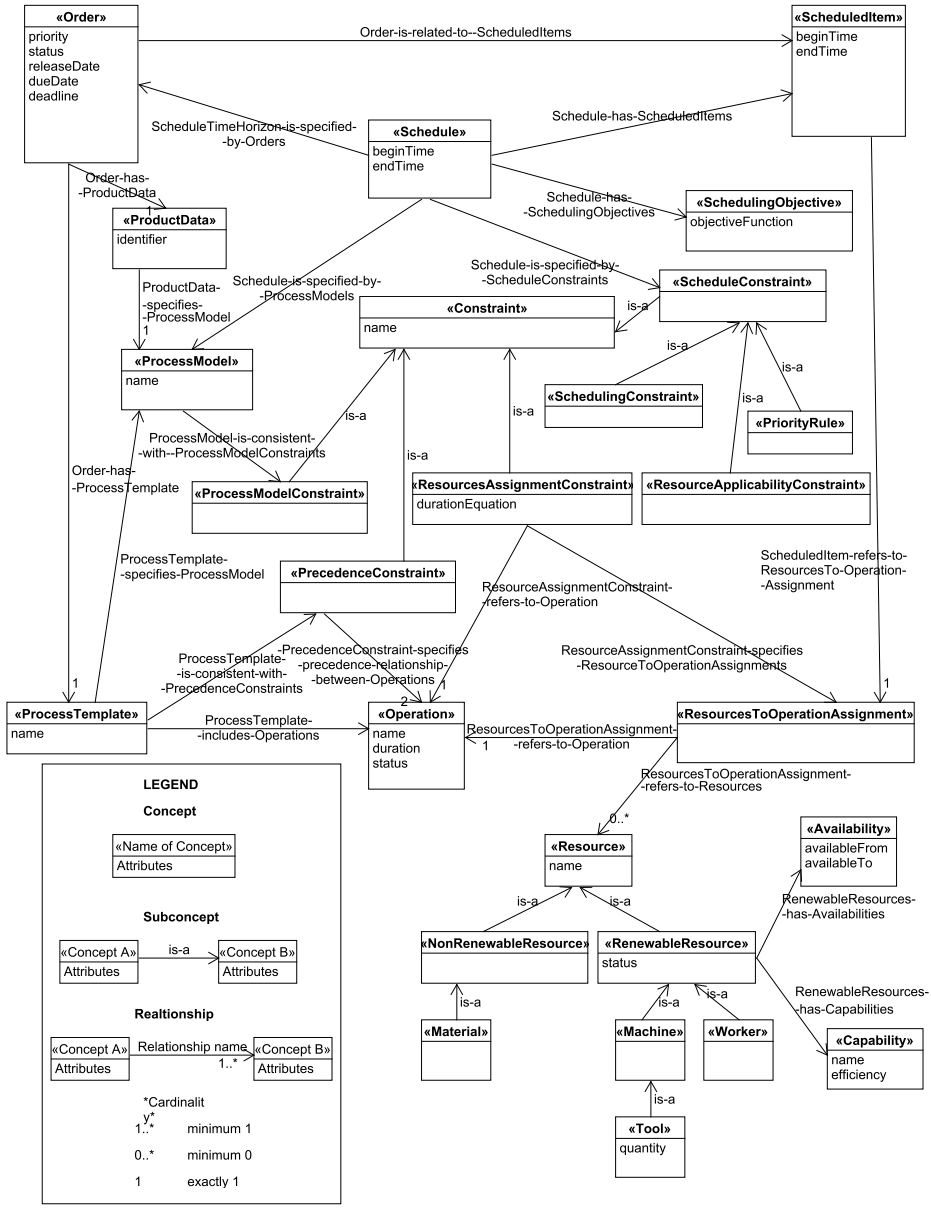

Figure 1. Main concepts of scheduling ontology

ProcessTemplate. A `ProcessTemplate` describes a technological workflow (sequence of `Operations`) required for the manufacturing of a particular type of a product. A `ProcessTemplate` should be specified for each `Order`. A `ProcessTemplate` is a pair $\langle \mathsf{O}, \mathsf{C} \rangle$, where O is a set of `Operations` and C is `PrecedenceConstraints` on it.

ProcessModel. A `ProcessModel` describes a unique and specific sequence of processing steps (`Operations`) that must be performed so that the ordered specific *product* is manufactured. A `ProcessModel` is specified by a `ProcessTemplate` and `ProductData`. There is exactly one `ProcessModel` for each `Order`. Therefore, there can be different `ProcessModels` for the same `ProcessTemplate`.

Schedule. A `Schedule` is a plan for the manufacturing of the ordered *products* (i.e., a fulfilment of the `Orders`). A `Schedule` is described by the *starting* (*beginDate*) and *finishing* (*endDate*) times of each `Operation` related to the particular `Order` considering the scheduling task and the particular `Resources` allocated to the `Operation` at this time range. In particular, a `Schedule` is a set of `ScheduledItems`, where a `ScheduledItem` is a quintuple of the following form:

$$\langle \texttt{Operation}, \texttt{Resource(s)}, \textit{beginDate}, \textit{endDate} \rangle.$$

A `Schedule` is *complete*, if for each `Operation` related to the particular `Order` to be scheduled there exists a unique `scheduledItem` in the `Schedule`. A `Schedule` is *correct* if each `ScheduledItem` has only one allocated time interval $[\textit{beginDate}, \textit{endDate})$ and if, in addition, it meets objectives of the scheduling (`SchedulingObjectives`) and `ScheduleConstraints`. The `SchedulingObjectives` may be different and should be specified by the user.

Constraint. `Constraints` define properties, rules or specific restrictions that must be satisfied. We distinguish four main basic types of `Constraints`:

1. `ProcessModelConstraints` define the principles for the construction of a `ProcessModel` for each `Order` and for the scheduling problem instance,
2. `PrecedenceConstraints` specify particular relationships between `Operations`. For example, the best known type of precedence relationships is the *finish–start relationship with a zero time-lag* ([13], pg.13),
3. `ResourcesAssignmentConstraints` represent restrictions, such as the minimal and maximal number of required `Resources` and specify all allowed assignments of `Resources` to the `Operation` (*ResourcesAssignmentGroups*) — that is, define for each `Operation` a set of particular `Resources` that are combined on the base of their particular `Capabilities`, such as specific skills of workers and/or functions of machines, or define a particular *name* of the `Resource` (e.g., name of worker),
4. `ScheduleConstraints` are rules that specify the `Schedule` (i.e., a sequencing of `ScheduledItems` of a `Schedule`) and check if `Resources` are assigned to `Operations` correctly.

4. Ontology-Based Architectural Solution of a Manufacturing Scheduling System

The main goal of the paper is to provide a manufacturing scheduling ontology based solution for developing easily customizable scheduling systems for small and medium size enterprises. For that purpose, the overall architecture of a scheduling system is proposed to be ontology aware by getting its main input from the manufacturing scheduling ontology and its individuals (instances).

4.1. General View of an Architecture and its Components

The proposed ontology based generic top-level architecture (see Figure 2) is composed of the *User Interface*, *Customized manufacturing scheduling ontology*, *SPARQL query engine*, *Scheduler* and *Schedule* modules. The functionality of each of these modules is described in more details as follows.

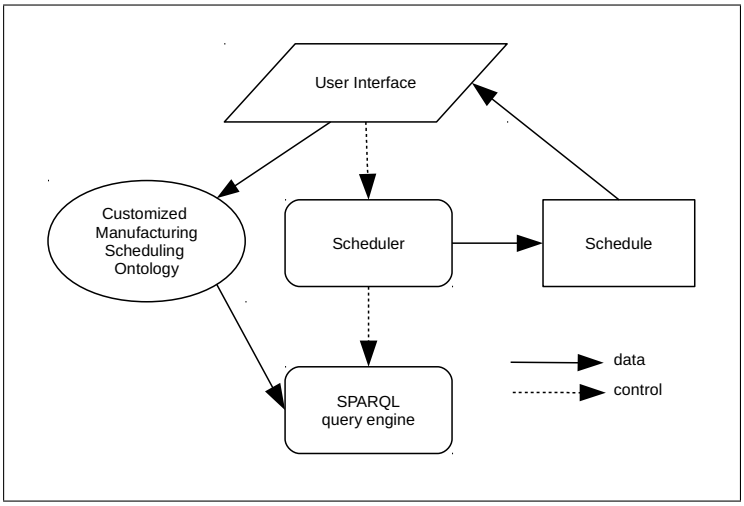

Figure 2. The main modules of the customized manufacturing scheduling system

User Interface. The *User Interface* module serves as an entry point of the manufacturing scheduling system. The usage of the manufacturing scheduling system always starts with the customization of the manufacturing scheduling ontology by defining the actual products, orders, resources and the manufacturing details. Scheduling can be started when the customized manufacturing scheduling ontology is complete. During its execution the *Scheduler* uses the *SPARQL query engine* to retrieve needed information from the *customized manufacturing scheduling ontology* and generates the *Schedule*. The *Schedule* is presented to the user and can be (re)used when executing the *Scheduler* next time in order to preserve some part of the previously generated schedule.

Scheduler. The *Scheduler* module is the core of the scheduling software that generates `Schedules` (the plans for processing `Operations` related to the `Orders` to be scheduled). The *Scheduler* module's scheduling process can be driven by the user by specifying the *scheduling strategy*, which defines the way of adding `Operations` in the case of gradual construction of the `Schedule` (e.g., a forward-, a backward or a multi-pass way) and a *time horizon* of the `Schedule` (i.e., the *begin*- and the *endDate* of the scheduling process). The *Scheduler* module is developed strictly based on the *manufacturing scheduling ontology*. Having the *manufacturing scheduling ontology* at hand the development of the *Scheduler* module can be done semi-automatically as is described in Ojamaa et al. ([14]) and Haav et al. ([15]).

Customized Manufacturing Scheduling Ontology. The *customized manufacturing scheduling ontology* extends the *manufacturing scheduling ontology* by adding actual individuals of `Product`, `Orders`, `Operations`, `Resources` and manufacturing related concepts (i.e., actual data required for the scheduling process). The *customized manufacturing scheduling ontology* is further discussed in the next subsection.

SPARQL Query Engine. The *SPARQL query engine* module enables queries from the *customized manufacturing scheduling ontology* to retrieve information that is stored in the ontology.

4.2. Customization of the Manufacturing Scheduling System Using the Formalized Manufacturing Scheduling Ontology

The manufacturing scheduling system provided in this paper can be easily customized to serve a new manufacturing process by utilizing the customized manufacturing scheduling ontology as an input to the system.

Customization of the manufacturing scheduling ontology basically means that when creating the customized manufacturing scheduling ontology the manufacturing scheduling ontology is imported to it and necessary individuals that are specific to the particular manufacturing scheduling process are created and asserted to the corresponding manufacturing scheduling ontology classes. In addition necessary object and data property assertions should be added to the customized manufacturing scheduling ontology. It is important to note that according to this method, namespaces of original manufacturing ontology and customized ontology are different. This makes ontologies modular and handling versioning and reuse of the original manufacturing ontology easy.

The provided original manufacturing scheduling ontology is designed so that linking properties of a particular product or product family data is easy. The manufacturing scheduling ontology has the class `ProductData` (see Section 3 and Figure 1). Its subclasses and datatype properties are not specified, which allows in the customized manufacturing ontology to add needed subclasses and properties according to the specific*product catalogue* that is used in the manufacturing process. In the cases that product data is located in relational databases or come from other resources it is possible to enhance the customized manufacturing scheduling ontology so that this information can be retrieved from these external resources by integrating these resources to the customized manufacturing scheduling ontology. In principle, one can add any specific class or property to the customized manufacturing scheduling ontology if necessary.

From the technical point of view the original manufacturing scheduling ontology and the customized manufacturing scheduling ontology are represented in OWL[5] that is widely used in semantic web technological space and supported by well-known standards by W3C. OWL ontologies are serialized in the Resource Description Framework (RDF)[6]. The scheduling system uses SPARQL[7] for retrieval of necessary information from the customized manufacturing scheduling ontology.

[5]https://www.w3.org/TR/owl2-primer/
[6]https://www.w3.org/TR/sparql11-query/
[7]https://www.w3.org/TR/sparql11-query/

In the following example, we demonstrate how the customized manufacturing scheduling ontology is created and used for the design of an experimental manufacturing scheduling system at one of our collaborators Bolefloor Ltd.[8]

In this paper, ontology fragments are given in OWL functional style syntax,[9] where the prefix "*schedule*" denotes the original manufacturing scheduling ontology elements and the prefix "*custschedule*" denotes the customized manufacturing scheduling ontology elements.

In Figure 3, a set of individuals of the customized manufacturing scheduling ontology is declared and a fragment of a set of declarations of classes, data and object properties of the original manufacturing scheduling ontology (see also Section 3) is provided. In the example, only six classes are declared as follows: `Resource`, `RenewableResource`, `Operation`, `Order`, `ProductData`, and `ProductTemplate`.

Declaration of Individuals

```
Declaration( NamedIndividual( custschedule:R_1 ) )
Declaration( NamedIndividual( custschedule:R_2 ) )
Declaration( NamedIndividual( custschedule:P_1 ) )
Declaration( NamedIndividual( custschedule:OP_1 ) )
Declaration( NamedIndividual( custschedule:OP_2 ) )
Declaration( NamedIndividual( custschedule:PT_1 ) )
Declaration( NamedIndividual( custschedule:O_1 ) )
```

Declaration of Classes

```
Declaration( Class( schedule:Resource ) )
Declaration( Class( schedule:RenewableResource ) )
Declaration( Class( schedule:Operation ) )
Declaration( Class( schedule:Order ) )
Declaration( Class( schedule:ProductData ) )
Declaration( Class( schedule:ProcessTemplate ) )
```

Declaration of ObjectProperties

```
Declaration( ObjectProperty( schedule:Order_has_ProductData) )
Declaration( ObjectProperty( schedule:Order_has_ProcessTemplate ) )
Declaration( ObjectProperty( schedule:ProcessTemplate_includes_Operations) )
```

Declaration of DataProperties

```
Declaration( Datatype( schedule:hasResourceName ) )
Declaration( Datatype( schedule:hasOperationName) )
Declaration( Datatype( schedule:hasOperationDuration ) )
Declaration( Datatype( schedule:hasOrderNumber) )
Declaration( Datatype( schedule:hasProductDataProductIdentifier ) )
```

Figure 3. Declarations of the customized manufacturing scheduling ontology individuals and the manufacturing scheduling ontology classes (a fragment)

A subset of axioms of the original manufacturing scheduling ontology are presented in Figure 4. These cover *subclass*, *datatype* and *object property axioms*.

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

[9] http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

Subclass axioms

SubClassOf(schedule:Order schedule:Thing)
SubClassOf(schedule:ProductData schedule:Thing)
SubClassOf(schedule:Operation schedule:Thing)
SubClassOf(schedule:ProcessTemplate schedule:Thing)
SubClassOf(schedule:Resource schedule:Thing)
SubClassOf(schedule:RenewableResource schedule:Resource)

DataProperty axioms

DataPropertyDomain(schedule:hasResourceName schedule:Resource)
DataPropertyRange(schedule :hasResourceName xsd:string)
DataPropertyDomain(schedule:hasOperationName schedule:Operation)
DataPropertyRange(schedule:hasOperationName xsd:string)
DataPropertyDomain(schedule:hasOperationDuration schedule:Operation)
DataPropertyRange(schedule:hasOperationDuration xsd:integer)
DataPropertyDomain(schedule:hasOrderNumber schedule:Order)
DataPropertyRange(schedule:hasOrderNumber xsd:string)
DataPropertyDomain(schedule:hasProductDataProductIdentifier schedule:Product)
DataPropertyRange(schedule:hasProductDataProductIdentifier xsd:string)

ObjectProperty axioms

ObjectPropertyDomain(schedule:Order_has_ProductData schedule:Order)
ObjectPropertyRange(schedule:Order_has_ProductData schedule:ProductData)
ObjectPropertyDomain(schedule:Order_has_ProcessTemplate schedule:Order)
ObjectPropertyRange(schedule:Order_has_ProcessTemplates schedule:ProcessTemplate)
ObjectPropertyDomain(schedule:ProcessTemplate_includes_Operations schedule:ProcessTemplate)
ObjectPropertyRange(schedule:ProcessTemplate_includes_Operations schedule:Operation)

Figure 4. Axioms of the manufacturing scheduling ontology (a fragment)

Assertion of individuals

ClassAssertion(schedule:Resource custschedule:R_1)
ClassAssertion(schedule:Resource custschedule:R_2)
ClassAssertion(schedule:ProductData custschedule:P_1)
ClassAssertion(schedule:Operation custschedule:OP_1)
ClassAssertion(schedule:Operation custschedule:OP_2)
ClassAssertion(schedule:ProcessTemplate custschedule:PT_1)
ClassAssertion(schedule:Order custschedule:O_1)

ObjectProperty assertions

ObjectPropertyAssertion(schedule:Order_has_ProductData custschedule:O_1 custschedule:P_1)
ObjectPropertyAssertion(schedule:Order_has_ProcessTemplate custschedule:O_1 custschedule:PT_1)
ObjectPropertyAssertion(schedule:ProcessTemplate_includes_Operations custschedule:PT_1 custschedule:
OP_1)
ObjectPropertyAssertion(schedule:ProcessTemplate_includes_Operations custschedule:PT_1 custschedule:
OP_2)

DataProperty assertions

DataPropertyAssertion(schedule:hasResourceName custschedule:"CNC1"^^xsd:string)
DataPropertyAssertion(schedule:hasResourceName custschedule:"CNC2"^^xsd:string)
DataPropertyAssertion(schedule:hasOperationName custschedule:"Brushing"^^xsd:string)
DataPropertyAssertion(schedule:hasOperationDuration custschedule:"10"^^xsd:integer)
DataPropertyAssertion(schedule:hasOperationName custschedule:"Calibrating"^^xsd:string)
DataPropertyAssertion(schedule:hasOperationDuration custschedule:"15"^^xsd:integer)
DataPropertyAssertion(schedule:hasOrderNumber custschedule:"34"^^xsd:string)
DataPropertyAssertion(schedule:hasProductDataProductIdentifier custschedule:"C8U"^^xsd:string)

Figure 5. Assertions of individuals, object properties, and data properties to the customized manufacturing scheduling ontology (a fragment)

In principle, these assertions provide customization of the original manufacturing scheduling ontology to meet needs of the particular manufacturing process. After input of assertions the customized manufacturing scheduling ontology is ready to serve the scheduling system. This is done by using SPARQL queries. The results of SPARQL queries are automatically inserted into the internal structure of the scheduling module.

For example, in the following Figure 6 SPARQL query is used to retrieve `Operation` names and the corresponding *durations*. Note that the query results are based on assertions given in Figure 5 that represent the corresponding data.

An example of the SPARQL query

```
PREFIX custschedule: <http://www.ioc.ee/customized_scheduleontology/0.1/>
PREFIX schedule: <http://www.ioc.ee/scheduleontology/0.1/>

SELECT ?hasOperationName ?hasOperationDuration
WHERE { ?x schedule:hasOperationName ?hasOperationName  ?x schedule: hasOperationDuration
?hasOperationDuration }
```

Query result

hasOperationName	hasOperationDuration
Brushing	10
Calibrating	15

Figure 6. An example of the SPARQL query for finding `Operations` and their *durations*

5. Experimental Implementation of the Architecture

We have experimentally implemented the ontology-driven scheduling system for orders-oriented and lean mass customization based manufacturing in cooperation with Bolefloor Ltd in the domain of production of wooden surface coverings [4]. This system was implemented basically in Java using the CoCoViLa [10] (Compiler Compiler for Visual Languages) model-based software development platform for implementation of graphical user interfaces and the composition of the whole system. In that experiment we have used only informal manufacturing scheduling ontology in order to drive the relevant architectural components of the scheduling system. We have put a lot of effort to the development of this ontology but did not formalize it to be used as a knowledge and data access point for the system. By now, we have changed the system architecture to utilize the manufacturing scheduling ontology developed in OWL and we are in the process of refactoring of the previous implementation according to the ideas given in this paper.

The main argument for re-design of the previous architecture and for the usage of semantic technologies was the need to easily adapt the manufacturing scheduling system

[10] http://www.cs.ioc.ee/cocovila/

to new product families and manufacturing processes. In addition, we have faced the problem that input data to the *InputData* module was given in spreadsheet format (e.g. MS Excel) that does not capture any semantics of given data.

In order to make the system more flexible, domain knowledge aware and easily customizable we have decided to develop the customization approach provided in this paper. Using the customized manufacturing scheduling ontology as an input to the system makes it possible to process data as well as semantic descriptions of data.

The architecture of the previous system was composed of the *Data Input* modules (*InputData*, *ProcessTemplates*, *SchedulingObjectives* and *ScheduleConstraints*), *Data Transformation* modules (*SchedulerData Modeler*, and *ProcessModeler*), *Scheduler* module and *Data Presentation module* (*Schedule*). The output schedules were generated in the form of Gantt charts. More detailed information of the previous system implementation can be found in [4].

6. Conclusion and Future Work

In this paper, we provided the manufacturing scheduling ontology and ontology-driven approach of design and development of manufacturing scheduling system for order-oriented and lean mass customization based manufacturing. The paper described the customization process of manufacturing scheduling systems that is based on utilization of the customized manufacturing scheduling ontology used as a common access point to the scheduling knowledge and data of the particular product manufacturing process. Using OWL for ontology representation has the following benefits: the customized manufacturing ontology is described using well-known standards of W3C, it can be retrieved in distributed environments using SPARQL and validated using Description Logic reasoners.

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd ed., 2008.

[2] S. Borgo and P. Leitão, Foundations for a core ontology of manufacturing, in *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, 751–775, Boston, MA: Springer US, 2007.

[3] S. Smith and M. Becker, An ontology for constructing scheduling systems, in *Working Notes from 1997 AAAI Spring Symposium on Ontological Engineering*, 120–129, AAAI Press, 1997.

[4] J. Sanko and V. Kotkas, Ontology - driven scheduling system for manufacturing, *Baltic Journal of Modern Computing*, **4** (3), 508–522, 2016.

[5] S. Smith, D. Hildum, and D. Crimm, Interactive resource management in the COMIREM planner, in *Proceedings IJCAI-03 Workshop on Mixed-Initiative Intelligent Systems*, 2003.

[6] D. Rajpathak, E. Motta, and R. Roy, A generic task ontology for scheduling applications, *Proceedings of the International Conference on Artificial Intelligence*, 2001.

[7] P. Leitao, A. Colombo, and F. Restivo, ADACOR: A collaborative production automation and control architecture, *IEEE Intelligent Systems*, **20** (1), 58–66, 2005.

[8] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider, DOLCE: a descriptive ontology for linguistic and cognitive engineering, *WonderWeb Project, Deliverable D17 v2*, **1**, 2003.

[9] S. Lemaignan, A. Siadat, J. Dantan, and A. Semenenko, MASON: A proposal for an ontology of manufacturing domain, in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)*, 195–200, 2006.

[10] A. Ojamaa, V. Kotkas, M. Spichakova, and J. Penjam, Developing a lean mass customization based manufacturing, in *16th IEEE International Conference on Computational Science and Engineering, CSE 2013, December 3-5, 2013, Sydney, Australia*, 28–33, 2013.

[11] R. Schonberger, Japanese production management: An evolution — With mixed success, *Journal of Operations Management*, **25** (2), 403–419, 2007.

[12] M. Pinedo, *Planning and Scheduling in Manufacturing and Services*. Springer Science and Business Media, 23rd ed., 2009.

[13] E. Demeulemeester, *Project Scheduling: A Research Handbook*. New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers, 2002.

[14] A. Ojamaa, H-M. Haav, and J. Penjam, Semi-automated generation of DSL meta models from formal domain ontologies, in *Model and Data Engineering*, 3–15, Springer International Publishing, 2015.

[15] H-M. Haav, A. Ojamaa, P. Grigorenko, and V. Kotkas, Ontology-based integration of software artefacts for DSL development, in *On the Move to Meaningful Internet Systems: OTM 2015 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, EI2N, FBM, INBAST, ISDE, META4eS, and MSC 2015, Rhodes, Greece, October 26–30, 2015. Proceedings*, 309–318, Springer International Publishing, 2015.