Abstract. This paper presents an approach of scheduling software development for orders-oriented and lean mass customization based manufacturing. To describe the various concepts of the manufacturing scheduling domain and their relationships an ontology for the scheduling domain is proposed. The scheduling software development and top-level architecture of the software are driven by this ontology.

The software is targeted at small and medium sized enterprises to solve their resource-constrained scheduling problems and fit well to their manufacturing process, allowing easy definition of new products and their production management. The software includes the customized scheduling algorithm for optimization of assigning of resources to operations and visual representation of workflow of manufacturing processes. The scheduling system for manufacturing is implemented in the CoCoViLa system.

Keywords: ontology, manufacturing scheduling, algorithm, software development

1 Introduction

Scheduling is a decision-making process that plays a crucial role in most manufacturing, production, and transportation systems, as well as in information processing systems, communication and other types of service industries.

Scheduling deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives (Pinedo, 2008). 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

* This research was supported by the Estonian Ministry of Research and Education institutional research grant no. IUT33-13.
Scheduling has been studied in detail by several authors (see, for instance, Pinedo (2009), Brucker (2001), etc). The main focus of research is on developing of specific applications and algorithms. Because of different scheduling objectives and additional constraints, such as priorities, sequent dependent set-up times or parallel resources, there is a huge number of scheduling problem classes. A widely used classification scheme of scheduling problems can be found in Brucker (2001). For each class of scheduling problems, sophisticated scheduling algorithms have been developed (see, for instance, Brucker (2001)).

We consider a scheduling problem as a constrained optimization problem, where time-constrained resources should be assigned to time-constrained operations at a particular time within a predefined time horizon of a schedule in accordance with predefined constraints and scheduling objectives.

The main contribution of the paper is the scheduling domain ontology-driven approach of design and development of scheduling software for order-oriented and lean mass customization based manufacturing. The software includes the following novel features: scheduling system architecture, visual representation of workflow of manufacturing processes and customized scheduling algorithm. For that we have elaborated a scheduling ontology that meets the requirements of order-oriented and lean mass customization based manufacturing.

The goal is to get rich and easily customizable scheduling system that could be exploited in different manufactures. Software elaboration and evaluation have been done in collaboration with Bolefloor Ltd that produce novel wooden design products (floors, furniture plates etc.) made of boards with naturally curved edges.

The rest of the paper is structured as followed. Section 2 is devoted to the related work and Section 3 provides an overview of the main concepts of scheduling ontology used for the order-oriented manufacturing systems and their relationships. Section 4 gives an overview of the system top-level architecture, main modules supporting the proposed approach, and the scheduling algorithm. Section 5 concludes the paper.

2 Related Work

2.1 Scheduling Systems in Manufacturing

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 with only small 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, 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

\[^{1}\text{http://www.bolefloor.com/}\]
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, ASPROVA (an Advanced Planning and Scheduling (APS) system). Our current experience shows that introduction and maintenance of these scheduling systems into an operational (small-scale) manufacturing needs too much effort. For detailed information see an overview and examples of scheduling systems in Pinedo (2008).

There have also been some attempts to develop ontologies for scheduling.

### 2.2 Scheduling Ontologies

*Ontology* is considered as a framework (a model building tool) for specifying models in a particular problem domain – that is, the scope of an application domain, various entities (concepts) in this domain along with their properties (attributes) or desired system functionalities and features.

The most known ontology that can be considered a classical scheduling ontology and provides a reusable and extensible base of concepts for specifying and representing constrained-based scheduling models for a range of applications in manufacturing, transportation logistics, etc is the OZONE ontology (Smith et al., 1997). The OZONE ontology provides a model of scheduling tasks, which are defined in terms of 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 a system COMIREM (Smith et al., 2003). 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 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. (2001) 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. (2001) 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.

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 (Ojamaa et al., 2013), the goal of
which is the mass product of unique and personalized products and elimination of the waste from the manufacturing.

3 Ontology for Scheduling of Manufacturing Systems

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 (Pinedo, 2009).

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: 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 Fig. 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 sake 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 that describes the product by defining several user-defined attributes, such as type (or production class), material, quantity, size.
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 — 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 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 Precedence-Constraints.

**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 \( \langle O, C \rangle \), where \( O \) is a set of Operations and \( C \) is a set of PrecedenceConstraints on it.

Diagrammatically, a ProcessTemplate can be described by means of the "activity-on-node" representation (see, for instance, Pinedo (2009, Chap. 4)), which uses a set of nodes to denote a set of Operations from within a ProcessTemplate and a set of arcs to represent a set of PrecedenceConstraints between these Operations. Thereby, each Operation in the ProcessTemplate is represented by a node and each directed arc is the symbolic representation of a PrecedenceConstraint between two distinct Operations.

For example, let us have a manufacturing process (see Fig. 2) consisting of five technological Operations \( O_3, O_4, O_5, O_6 \), and two dummy Operations \( O_1 \) (the begin Operation) and \( O_7 \) (the end Operation). To process Operation \( O_3 \), Operation \( O_2 \) has to be finished, to process Operation \( O_6 \), Operation \( O_5 \) and Operation \( O_4 \) either can be processed in parallel or in any order and so on.

![Fig. 2. Example of a ProcessTemplate](attachment:fig2.png)

Formally, this ProcessTemplate is defined by the following:

\[
\text{ProcessTemplate} = \langle \{O_1, O_2, O_3, O_4, O_5, O_6, O_7\},
\{C_{23} : \langle O_2 \lessdot O_3 \rangle, \\
C_{34} : \langle O_3 \lessdot O_4 \rangle, \\
C_{35} : \langle O_3 \lessdot O_5 \rangle, \\
C_{46} : \langle O_4 \lessdot O_6 \rangle, \\
C_{56} : \langle O_5 \lessdot O_6 \rangle \} \rangle,
\]
where $C_{23}, C_{34}, C_{35}, C_{46},$ and $C_{56}$ are PrecedenceConstraints.

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 Product-Data. 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 fulfillment of the Orders). A Schedule is described by the starting ($beginTime$) and finishing ($endTime$) 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

$(\text{Operation}, \text{Resource(s)}, beginTime, endTime)$.

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 $[beginTime, endTime]$ 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 (Demeulemeester (2002, 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-Driven Scheduling System Description

In this section we shortly describe the CoCoViLa tool used for the software development, top-level architecture of the scheduling system, its main modules and how these relate to the manufacturing scheduling ontology presented in the previous section.
4.1 CoCoViLa - a model-based software development tool

CoCoViLa\(^2\) (Compiler Compiler for Visual Languages) is a model-based software development platform that provides a framework for developing software. The CoCoViLa tool supports convenient implementation of components, preferably visual specification of software models and automatic code generation from a model. CoCoViLa includes two visual editors — the Class Editor for developing domain-specific concepts and the Scheme Editor for the visual composition of software models — the specifications. The core functionality of CoCoViLa is achieved by the usage of the Synthesizer that automatically generates Java programs from the specifications. CoCoViLa is being developed in the Software Science Laboratory of the Institute of Cybernetics at TUT.

CoCoViLa allows visual representation of specifications. A concept together with its visual representation specified in CoCoViLa is called a visual class. Visual classes are used to compose schemes in the CoCoViLa (Scheme Editor). For each visual class the following components are defined: Java class, visual image, set of ports (ports indicate which components of the class can be visually connected to other components in a scheme), icon image (a small raster picture that is shown on a toolbar) and a specification. An executed program can show results both in a separate window or display the feedback directly to the scheme.

The Java class together with the specification is called a metaclass, where the specification, also called a metainterface, is presented as formulas – axioms with realisations given by methods of its Java class. These formulas constitute a theory in intuitionistic logic that is used by structural synthesis of programs (Mints et al., 1982) for automatic construction of programs in CoCoViLa.

Thus, in CoCoViLa, a software package is a collection of visual classes and schemes related to an application domain. A description of a package is stored in the XML-based format. Each package can have its own domain-specific visual specification language (DSVL). See details of the CoCoViLa system in Grigorenko (2010) and Penjam et al. (2015).

4.2 The Main Modules of the Scheduling System

The ontology for scheduling of manufacturing systems defines the modular structure of the architecture of the scheduling system and relationships between modules. The core of the scheduling system is the Scheduler module that performs the scheduling task. From the manufacturing scheduling ontology we can also derive the essential data dependencies, that is to generate a Schedule we need Orders, their ProcessModels, ScheduleConstraints and SchedulingObjectives — concepts that the Schedule "is-specified-by".

The generic top-level architecture (see Fig. 3) is composed of the Data Input modules (InputData, ProcessTemplates, SchedulingObjectives and ScheduleConstraints), Data Transformation modules (SchedulerData Modeler and ProcessModeler), Scheduler module (Scheduler) and Data Presentation module (Schedule).

Let us describe these main modules in more details.

\(^2\) http://www.cs.ioc.ee/cocovila/
4.2.1 Data Input Modules

InputData Module. The InputData module contains static data, which does not depend on the Schedule, and dynamic data, which is dependent on the Schedule. The static data includes resources data, such as machines- and workers data, and orders data, such as the ordered quantities, due dates, release dates, and the priorities (weights) of the orders. The dynamic data depends on the Schedule and consists of the starting and completion times of the operations, the idle and setup times of the machines, the number of operations that are late, and so on. The InputData module is customer-specific.

ProcessTemplates Module. The ProcessTemplates module comprises a "library" of ProcessTemplates that describe the technological workflows required for the manufacturing of products. To compose a ProcessTemplate, an additional system for composing ProcessTemplates visually is used.

If needed, a ProcessTemplate can be modified by adding or excluding some Operations and so creating a new ProcessTemplate that corresponds to the new technological workflow required for the manufacturing of the particular type of a product.

SchedulingObjectives Module. The SchedulingObjectives module allows the user to specify various scheduling objectives like minimize a makespan of each Order or minimize a makespan of the Order with higher priority or minimize the number of Orders completed after their due date or minimize the number of Orders completed after their deadline or use Resources in efficient way or maximize the loading of the specific Resource.

ScheduleConstraints Module. The ScheduleConstraints module allows the user to specify a Schedule by defining the PriorityRules, which specify a priority in which a particular Operation related to the Order should be scheduled. For example, the
following Priority Rules can be specified: the Earliest DueDate First (EDDF), the
Highest Priority First (HPF), Sort In Random Order (SIRO).

4.2.2 Data Transformation Modules

ProcessModeler Module. The ProcessModeler module generates a ProcessModel for
all Orders to be scheduled (i.e., for the scheduling problem instance), knowing the
ProcessTemplates and ProductData (data related to the ordered product such as
size or quantity) associated to each Order to be scheduled.

SchedulerData Modeler Module. The SchedulerData Modeler module converts data
from InputData module (operations data, process templates data, resources data, such as
machine- and worker data, and order data, such as ordered quantities, due dates, release
dates, the priorities (weights) of the orders, etc) into data required for the scheduling
process.

4.2.3 Scheduler Module

Scheduler Module. The Scheduler module generates Schedules (the plans for pro-
cessing Operations related to the Orders to be scheduled). The Scheduler module
allows the user to specify the scheduling process by selecting 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 endTime of the scheduling process). The Sched-
uler module is the core of the scheduling software, it is developed strictly based on the
manufacturing scheduling ontology. Having the manufacturing scheduling ontology at
hand the development of the Scheduler can be done semi-automatically as described
in Ojamaa et al. (2015) and Haav et al. (2015). The algorithm used in the Scheduler
module is presented in Section 4.3.

4.2.4 Data Presentation Modules

Schedule Module. The Schedule module presents Schedules in the form of Gantt
charts. The Gantt chart is the usual horizontal bar graph, where the horizontal axis
represents time, and the vertical axis represents various resources, such as machines or
workers, or orders. For example, if the horizontal bar represents an operation, then the
length of the bar corresponds to the time required to complete this operation.

The Schedule module allows the user to select the view of the Gantt chart. We use
two different kinds of Gantt charts: the resource-oriented (see Fig.4 for workers) and
the order-oriented.
4.3 Scheduling Algorithm

Different scheduling objectives and additional constraints, such as priorities, sequence dependent set-up times or parallel resources divide scheduling problems into a huge number of classes. A widely used classification scheme of scheduling problems and sophisticated scheduling algorithms that have been developed for each class of scheduling problems can be found in Brucker (2001).

In this subsection we describe a customized Scheduling Algorithm for constructing a feasible Schedule and optimizing for an assignment of Resources to Operations. The goal of our Scheduling Algorithm is to find a “good” feasible Schedule in accordance with SchedulingObjectives defined by the user. In particular, the goal is to minimize the number of Orders that are completed after their respective deadlines, where Resources and Operations assignment is optimised.

We assume that the pre-emption of any Operation is not allowed — that is, an Operation is processed in an uninterrupted mode and each Operation can be scheduled only after all its predecessor Operations (specified by ProcessModel) are completed and all Resources required for its processing are available and not assigned to other Operations. We assume that the same Resource can not be assigned to any two different Operations simultaneously. If any Resource is already assigned to some Operation in a Schedule (at a particular time range), then this Resource is unavailable for any other Operations at this particular time range and thus other relevant time periods must be generated for assigning to the ScheduledItem. A beginTime of the first ScheduledItem in the Schedule must be greater than or equal to a beginTime of a Schedule and the endTime of the last ScheduledItem of a Schedule must be less than or equal to the endTime of the Schedule.
The input for the Scheduling Algorithm is the scheduling problem instance provided by specification of a scheduling problem \((\text{schedulingProblemSpec})\) defined by the ProcessModel and the time range of the Schedule \((\text{scheduleTimeHorizon})\), and by SchedulerData — data that is required for the Scheduling Algorithm, such as Resources, ResourcesAssignmentGroups and Orders’ data (ordered quantities, deadlines, due dates, release dates, the priorities or weights) and by PriorityRules that are specified by the user. The output for the Scheduling Algorithm is a feasible Schedule (a list of ScheduledItems) for the given scheduling problem instance whenever one exists, or "no feasible schedule exists" if no feasible Schedule exists.

At the current version of the Scheduling Algorithm we apply a constructive forward scheduling approach that allows to gradually construct a Schedule by adding one ScheduledItem at a time in the increasing direction of time starting at the begin-Time of the Schedule. In forward scheduling, we use Earliest Deadline First (EDF) PriorityRule, i.e., among all Operations the Operation that has the earliest deadline will be scheduled first and Operations are selected arbitrarily, if there are multiple Operations with equal deadlines.

The Scheduling Algorithm has two main steps. The first step, initialization, includes setting the beginTime, the endTime, the currentTime of the Schedule, and the maximum number of iterations (Limit) of the Scheduling Algorithm (for the cases when the feasible Schedule is not to be found within reasonable period of time).

The second step, the while loop, repeatedly executes the following four sub-steps until a feasible Schedule is obtained (if such exists) or no feasible Schedule exists:

1. Search for all "ready" Operations with all their predecessor Operations scheduled and sort them according to the Earliest Deadline First (EDF) PriorityRule.
2. Select the Operation with the earliest deadline, find all combinations of Resources (ResourcesAssignmentGroups) required for its processing that are available and not assigned to any other Operations at the curTime of the Schedule and select the best one, for example select a ResourcesAssignmentGroup that comprises the minimal number of Resources or gives the minimal processing time. If no Resource can be assigned, then select the next "ready" Operation in the list.
3. Compute the duration of Operation on the basis of the number and efficiency of Resources assigned to it (we can not determine the duration of the Operation until we know which Resources are assigned to it) and create the corresponding ScheduledItem. Add the ScheduledItem to the Schedule. Repeat the sub-steps 2 and 3 for all "ready" Operations.
4. Search all ScheduledItems to find the one with the earliest endTime and deallocate all Resources assigned to the Operation, change the curTime of the Schedule to the endTime of the ScheduledItem. Repeat the sub-steps from 1 through 4.

Here follows a description of the Scheduling Algorithm in pseudo-code.
Algorithm 1: Scheduling Algorithm: ForwardStrategy

Input: schedulingProblemSpec(scheduleTimeHorizon, ProcessModel), PriorityRules, SchedulerData

Output: Schedule

1. Initialize the Schedule:
   1. Specify the begin-, the end- and the current times of the Schedule,
   2. and the maximum number of iterations of the Scheduling Algorithm (i.e., Limit);

4. While (there exists at least one Operation with a status "pending" or "running") and
   (the current time of the Schedule is less than the end time or the number of iterations of
   the Scheduling Algorithm is less than or equal to the specified Limit) do

   2.1 For all Operations of the ProcessModel with all their predecessor Operations
       having the status "completed" (completedOperations set) change their status to the
       "ready" and add to the readyOperations set
   2.2 Sort the readyOperations set according to the specified PriorityRule(s)
   2.3 foreach Operation in the readyOperations set do
       2.3.1 find all Resources required for its processing that are available and not in
           the status "busy" at the current time
       2.3.4 select the best (combination of) Resources according to the predefined
           Criterion (e.g., the minimal duration or minimal number of required
           Resources);
       2.3.5 compute the duration of the Operation;
       2.3.6 change the status of the Operation to the "running";
       2.3.7 change the status of each Resource assigned to the Operation to the
           "busy";
       2.3.8 create the corresponding ScheduledItem and add it to the Schedule
   2.4 among all ScheduledItems of the Schedule do
       2.4.1 find the ScheduledItem with the earliest endTime;
       2.4.2 change the current time of the Schedule to the endTime of the
           ScheduledItem;
       2.4.3 change the status of the corresponding Operation to the "completed";
       2.4.4 For each Resource assigned to the Operation change the status of the
           Resource to the "idle";

5 Conclusion and Future Work

This paper presents a description of the ontology-driven system for solving resource-constrained scheduling problems in orders-oriented and lean mass customization based manufacturing using the CoCoViLa system. This work has been done within the project Model Based Java Software Development Technology\(^3\) that assumes starting system development from the ontological conceptualization of the domain, presenting the (meta-)models of the problem and implementing the system via transformations of the models into code. The experiments on models of manufacturing processes and development of software components are done in collaboration with Bolefloor Ltd.

\(^3\) [http://cs.ioc.ee/mbjsdt/](http://cs.ioc.ee/mbjsdt/)
So far we have used a simplified model of the scheduling problem and as a consequence a simple Scheduling Algorithm is developed. In practice, the scheduling problem is more complex, where Operations may be processed for a period of time, interrupted and resumed at a later time, even by another Resource (a pre-emptive scheduling problem), a complicated setting of Resources and computation of durations of Operations are required.

One direction for the future research is elaboration of more sophisticated scheduling algorithms. We see two possible ways in doing it: further developing our own algorithm or incorporating some existing scheduling engine via defining an ontology for it.

Another direction for the future research involves creation of customer-specific ontology and merge with the manufacturing scheduling ontology. This enables development of generic Data Transformation modules for handling data of different customers. Currently Data Transformation modules are customer specific.

References


Received April 29, 2016, accepted April 30, 2016