Computer Science

Kjell Olofsson

An investigation into production scheduling systems

D-dissertation (10 p)

2004:06

This report is submitted in partial fulfillment of the requirements for the Master's degree in Computer Science. All material in this report which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Kjell Olofsson

Approved, 2004-12-09

Opponent: Thijs Holleboom

Advisor: Donald Ross

Examiner: Anna Brunström

Abstract

Production scheduling consists of the activities performed in manufacturing companies to manage and control the execution of the production process. The basic task is to perform the production as planned while at the same time trying to satisfy the overall goals of the company. This is an important part of the management of a company since it directly affects the performance of the enterprise. There exists much interest from industry in using software systems to support the scheduling process but the application of such systems has shown to be problematic.

This dissertation is an investigation into the area of production scheduling and production scheduling systems. The purpose of the study is to determine which requirements there are on a scheduling system and what functionality such a system should provide. The aim has been to maintain a practical focus and try to find requirements that are important in reality when a system is used in a company.

The investigation has been performed through literature studies and by performing a case study in a company that use a scheduling system. From the information gathered in the investigation, a design of a scheduling system framework has been proposed and a prototype of the most important parts of this framework has been implemented. The results so far show that scheduling systems satisfying the requirements elicited from the investigation can be developed using the proposed framework.

Contents

List of Figures

List of abbreviations

APS	Advanced Planning and Scheduling
BOM	Bill of Material
CSP	Constraint Satisfaction Problem
ERP	Enterprise Resource Planning
FCS	Finite Capacity Scheduling
J2EE	Java 2 Enterprise Edition
MES	Manufacturing Execution System
MPC	Manufacturing Planning and Control
MPS	Master Production Schedule
MRP	Material Requirements Planning
RCCP	Rough Cut Capacity Planning
SBP	Shifting Bottleneck Procedure
TOC	Theory of Constraint

1 Introduction

Production scheduling is the activities performed in manufacturing companies to manage and control the execution of the production process. The basic task is to perform the production as planned while at the same time trying to satisfy the overall goals of the company. Since how production scheduling is performed directly affects a manufacturing company's goals, such as customer service level and resource utilization, it ultimately determines the overall performance of the company. This makes it a very important area in a company's management.

Much effort has been put into developing methods and practices for the management and control of production processes both in industry and in academic disciplines such as industrial engineering, operational research and artificial intelligence. The management and control processes encompass activities beginning with demand management, followed by higher level planning and then production scheduling before performing the actual production. Many of the methods involve large amounts of information processing and therefore need to be supported by different kinds of information systems. These systems have proved successful for the higher levels of planning but when it comes to the scheduling activities, the use of computerized systems has shown to be more problematic. There are manufacturing companies that successfully utilize scheduling systems but there also exist many examples of failed implementations, and it appears to be difficult to apply theoretical scheduling methods in practice. Still, because the importance of the scheduling activities in a company, there is much interest from industry in such systems and potentially it should be possible to support and make the production scheduling process more effective by using appropriate software tools.

This dissertation is an investigation into the area of production scheduling and production scheduling systems and what functionality a scheduling system should provide. The study has been performed for a software company called ComActivity [9] which develops systems for manufacturing industries. The aim has been to find out what functionality that is required from a scheduling system that can be put to practical use in a production environment. Because of the apparent difficulties in applying scheduling theories in practice, an important aspect of the study has been to always maintain a practical focus and from that standpoint apply a more theoretical approach where it has seemed appropriate.

1.1 Production scheduling, a brief introduction

The basic task in production scheduling is to determine how production should be performed in a factory. From an aggregated higher level plan, a schedule is constructed that describes in detail which activities must be performed and how the factory's resources should be utilized to satisfy the plan. The resources can for example be machines and personnel that are needed to perform the activities. The schedule then describes in what order and by which resources the activities shall be performed.

Production scheduling is often a complex task with many factors to consider. There are often complicated precedence relationships between the activities and between the activities and the production resources. The available capacity of the production resources is limited and must therefore be used as effectively as possible. It is also common that the production environment contains high levels of uncertainty that adds to the complexity of the problem.

Traditionally, scheduling has been performed manually but in attempts to produce better schedules and remove tedious manual work, different kind of supporting software systems have been developed and used. These systems try to, by taking into account the different constraints that exist in the production environment, produce schedules that are realistic and satisfy the goals of the company.

1.2 ComActivity

ComActivity develops systems for manufacturing companies and provides software solutions for process- and workflow centered application development together with a runtime environment for the deployed solutions. The solutions are built in a J2EE [17] environment and use a service-oriented architecture. The solution encompasses tools for data-modeling as well as for design of process flow, workflow and user-interfaces and also includes a graphic scheduling tool with an interactive Gantt-chart[1]. The solution deployed is completely web-based and a service-oriented architecture allows for easy integration with other systems.

[1] A Gantt-chart is a diagram that, in its most common form, depicts activities on an x-y grid where the x-axis represents time and the y-axis resources. An activity is represented by a rectangle on the row representing the resource that performs the activity. The length of the rectangle along the x-axis corresponds to the time required to perform the activity.

1.3 The scope of this dissertation

The purpose of this dissertation is to find out what is required from a scheduling system in practical use and also how ComActivity's solution can be used to provide this. The first part of the dissertation, gives background information and investigates what production scheduling is, what the problems are and what theories that have been developed around it. To keep a practical focus, this part also includes a case study performed in a company that has used a scheduling tool for a couple of years. In the second part, a design of a scheduling system framework is described and a prototype of a component with core scheduling functionality is implemented.

1.4 Dissertation layout

The first part of the dissertation consists of chapter 2 to chapter 6. This part begins with some background information necessary to understand what production scheduling is. After that production scheduling is described in more detail and different aspects of the subject is investigated. Some scheduling theory is overviewed and different categories of software that are involved in scheduling activities are described. Part one ends with a description of the performed case study.

Chapter 2 puts production scheduling into context, by giving a general overview of production planning and control. Some required terminology is also explained in this chapter.

Chapter 3 describes production scheduling in more detail. The first part of this chapter describes important characteristics of production environments, scheduling objectives and the organization and information flow in a scheduling process. The second part begins with a section on how schedules can be compared and evaluated, followed by a description of different scheduling methods and finally the important aspect of scheduling in practice is discussed.

Chapter 4 contains a brief overview of the large research area of scheduling theory. The purpose of the chapter is to give an overview of the theoretical problem formulation and some solution methods that can potentially be used in practice.

Chapter 5 is an overview of software systems for manufacturing planning and control as well as scheduling.

Chapter 6 describes the case study. The case study has been performed in a company that uses a scheduling system. The chapter begins with descriptions of the studied production environment, the scheduling process in the environment and the scheduling objectives of the

company. After that the scheduling algorithm applied by the existing scheduling system is described. The chapter ends with a discussion of the study.

The second part of the dissertation consists of chapter 7 to chapter 10. In this part, the information gathered in part one is used to find requirements that should be satisfied by a scheduling system and from this a scheduling system framework is proposed. A prototype implementation of the most important parts of this framework is described. The proposed framework is evaluated and the results and conclusions from the investigation are discussed.

Chapter 7 describes the proposed scheduling system framework and the implemented prototype. The chapter begins with a description of the requirements on a scheduling system that has been elicited through the literature studies and the performed case study of part one. Next the design of the proposed framework is described and the chapter ends with a description of the implemented prototype.

Chapter 8 summarizes the results from the investigation. The chapter contains sections on the difficulties in the implementation and use of scheduling systems, the reasons for implementing a scheduling system, the application of advanced scheduling methods and lastly on the requirements on a scheduling system.

Chapter 9 is an evaluation of the proposed scheduling framework. The evaluation is performed by comparing the schedules produced by the prototype against the schedules produced by the system used in the environment described in the case study and by evaluating how the framework satisfies the requirements derived in part one of the dissertation.

Chapter 10 contains a conclusion and brief summary. The chapter ends with a section on future work and areas that should be investigated further.

2 Manufacturing planning and control from a scheduling perspective

Production scheduling is part of a process called manufacturing planning and control (MPC) which encompasses the activities performed in a company to plan and control its production, from initial demand management to execution of work on the shop floor. This chapter describes the role of scheduling in MPC and how that role has evolved and become more and more complex. Since some special terminology is used in the dissertation, a section explaining these expressions is included at the end of the chapter.

2.1 Introduction

The purpose of MPC is to determine what should be produced in a company and then to produce this as effectively as possible. The MPC activities must always be performed in a way that satisfies the overall goals of the company, typically to maximize profit and minimize cost, while at the same time making sure the production can be controlled in a practical way.

From some sort of strategic plan at a high level through planning and scheduling activities a number of what, where and when questions have to be answered which ultimately will result in a detailed schedule that can be executed by the company's production resources. Historically this has of course been done manually but since computers became available for more widespread use, MPC has been computerized and the evolution of MPC has very much followed the evolution of computers.

The most widely used method for this is called Material Requirements Planning (MRP). To put production scheduling in context, MRP is described in section 2.2.

2.1.1 The difference between planning and scheduling

The difference between planning and scheduling is a somewhat blurred area and the definition to some degree varies between different sources in the literature. The general idea is that planning is done at a higher, aggregated level over longer time periods and that scheduling involves more details and is done over shorter time periods. As the MPC activities proceed from planning to scheduling each step adds more details and brings the initial demands closer to being executed on the shop floor. Planning uses expected demands and forecasts and

therefore always contains some level of uncertainty. As the plans evolve through the planning and scheduling process and details are added, that uncertainty is gradually removed.

2.2 Material requirements planning

MRP has its origins in the 1950s and is a method whose initial goal was to enable more effective material handling [30]. The basic idea is, from demands at an aggregated level both in time and quantity, to step by step disaggregate and add more details and at last arrive at a detailed schedule that can be executed in the production environment. MRP was enabled by the more common availability of computers which made it possible to handle, and perform calculations on, large amounts of data.

2.2.1 A general model

The general model described here comes from [35] and is also the model used by APICS – The Educational Society for Resource Management in their certification programs [3]. APICS sets much of the de facto standards in the area of MPC and most vendors of MPC-software uses these standards as a basis for their systems.

MRP according to this model is grouped into three phases:

1. creating the overall manufacturing plan
2. detailed planning of material and capacity needs
3. execution of detailed plans

The processes involves developing aggregated plans in the first phase, disaggregating these plans and adding more details in phase two and finally executing the detailed plans on the shop floor and in the purchase department in phase three.

Figure 2.1 is adapted from [35] and shows the three phases and the information flow between them. In the figure, rectangles with continous lines represent activities and rectangles with broken lines represent the information that is communicated between, or used by, activities.

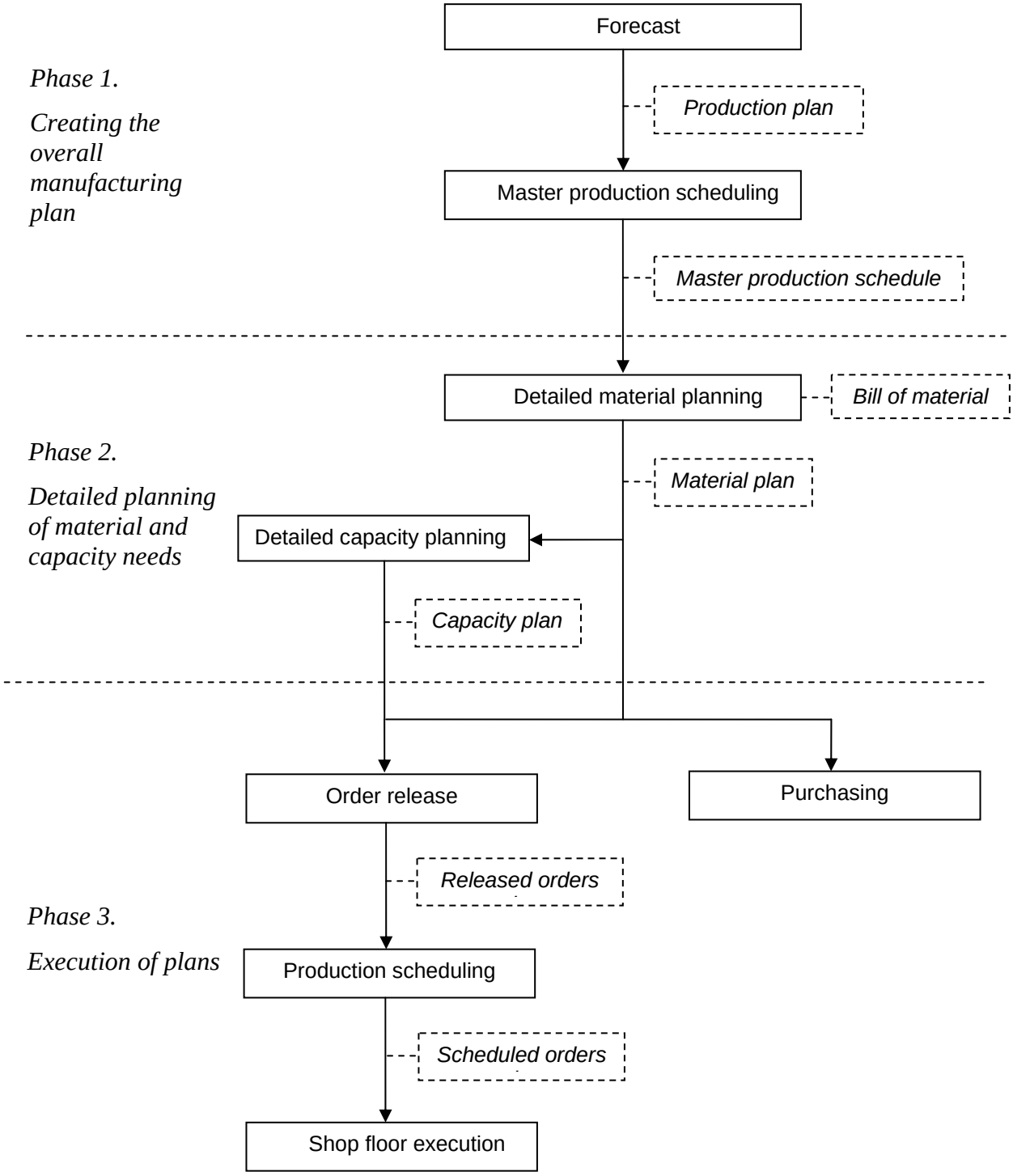


Figure 2.1. Model for material requirements planning.

2.2.1.1 *Phase 1, Creating the overall manufacturing plan*

The forecast is an estimate of the future demand for the company's products or services over some time period, usually 3 – 12 months. The forecast results in a production plan that contains aggregated information of what the company shall produce. The information in the production plan is often given on a per month basis. In master production scheduling the production plan is translated into production terms such as end items that can be used in phase two.

2.2.1.2 *Phase 2, Detailed planning of material and capacity needs*

The master production schedule (MPS) is the input to phase two where it is disaggregated into detailed, time-phased requirements of which items must be manufactured or purchased to satisfy the expected demands. To do this a list called the Bill of Material (BOM) containing all the raw materials and sub parts that make up the end items in the MPS is used. The current inventory status together with the BOM makes it possible to calculate the actual production and purchasing requirements. The detailed material plan together with routing information, describing in which order items must be manufactured, is then used to calculate the production capacity needed to accomplish the material plan. Detailed material and capacity plans are the output of phase two.

2.2.1.3 *Phase 3, Execution of the detailed plans*

This phase involves purchasing the raw materials and sub-parts required and releasing production orders to be scheduled and then executed on the shop-floor. The scheduling determines in which sequences and in which production resources the released orders should be executed. Production resources are personnel, machines and other equipment that is used to perform the production.

2.2.2 The evolution of material requirements planning

MRP has evolved over time into what is now called Enterprise Resource Planning (ERP). MRP's focus was initially on materials handling and has some shortcomings when it comes to actually performing the production. One reason for this is that the detailed capacity planning in phase 2 (section 2.1.1.2) calculates the capacity required to perform the MPS but does not consider what capacity is actually available. When then schedules are executed with the actual capacity available in the production resources of the shop, they are therefore often impossible to follow [31].

To improve the produced schedules MRP evolved so that information from the shop floor is fed back to the MPS in what is termed *closed loop MRP*. This in turn evolved into Manufacturing Resource Planning (MRP II) which refines the capacity management with Capacity Requirements Planning (CRP) and also integrates the company's financial management into the model. In MRP II the detailed material plans are checked against available capacity in what is called Rough Cut Capacity Planning (RCCP) to determine if they are possible to execute. If RCCP shows that it is not possible to perform the planned production the MPS is changed and a new detailed material plan is produced. The loop from MPS to detailed material plan to RCCP continues until a feasible plan is developed.

MRP II then has expanded into ERP which also includes, among other things, engineering and distribution activities but ERP does not contain any major changes in the planning and scheduling process [31].

2.2.3 Shortcomings of material requirements planning

Despite the improvements of MRP and the evolution into ERP, developing feasible schedules is still a problem. The capacity check that is performed with RCCP is too coarse to determine if capacity actually is available when needed and the material and capacity requirements are not synchronized with each other when making the check. This problem has been termed the vertical and horizontal separation of materials and resources [26]. The vertical separation means that planned production is separated from actual production and the horizontal separation mean that material and resource requirements are not synchronized. The separation problem means that since the feasibility of the resource and material requirements has been checked at an aggregated level, the actual detailed requirements of the orders that are released from phase 2 may well be unsynchronized, making it impossible to develop a feasible schedule.

Many of the reasons for these shortcomings go back to the initial development of MRP when computer capacity was scarce and one way to make such systems possible at all was to reduce details by using aggregated information and to separate problems into smaller parts that could be more easily processed. As the computer evolution went on more and more details have been added in MRP systems but the basic structure and methods are still much the same.

2.3 Terminology

Some terminology that is used in the rest of the dissertation is explained below. To provide for easier reading, the expressions are explained in a topical order.

Order - an order in this context is an ordered sequence of one or more operations that is required to be performed to produce some part, product or service. See Figure 2.2.

Job – synonym for order that is often used in scheduling literature

Operation - an operation is an individual activity or task that must be performed to fulfill an order. The operation must be performed by a specified resource or a set of alternate resources and requires a specified set of raw materials and/or sub-parts. See Figure 2.2.

Operation setup time – the preparation time needed before an operation can start.

Operation run time – the time it takes to perform the operation.

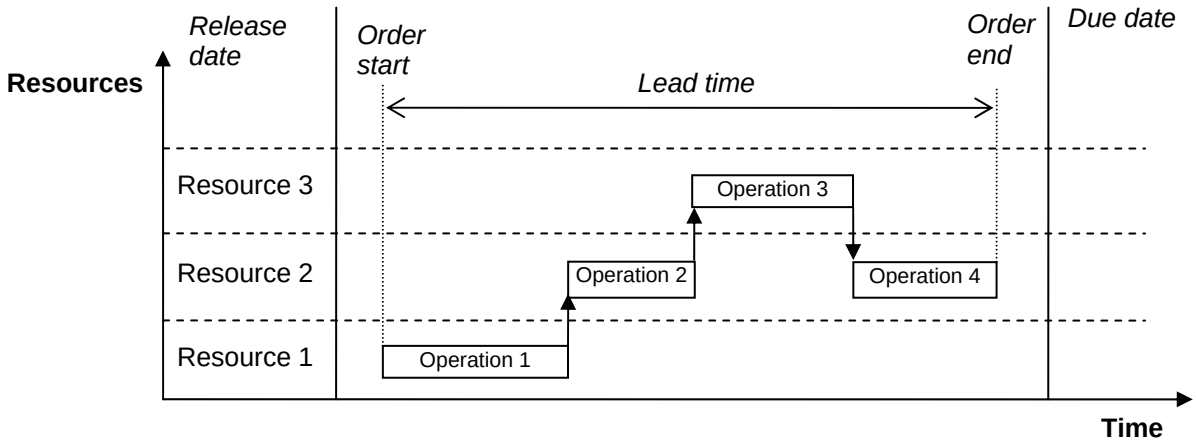


Figure 2.2. Order with operations.

Release date - the earliest date the first operation of the order can start.

Due date - the date when the order is planned to be finished.

Order start and *Order end* - the current start and end of the order when it is in process.

Lead time - the total time it takes to execute an order and is the time from the first operation is started to the last operation is finished.

Order slack – The sum of all setup and run times for all remaining operations subtracted from the time remaining to the due date

Lateness – the difference between an order's due date and the actual end date of its last operation. Lateness can be positive or negative, positive if the end date is after due date and negative if end date is before due date.

Tardiness – a tardy order is one who's last operations end date is after its due date. Tardiness is the same as positive lateness.

Forward scheduling - operations of an order are scheduled as early as possible starting from the release date. See Figure 2.3.

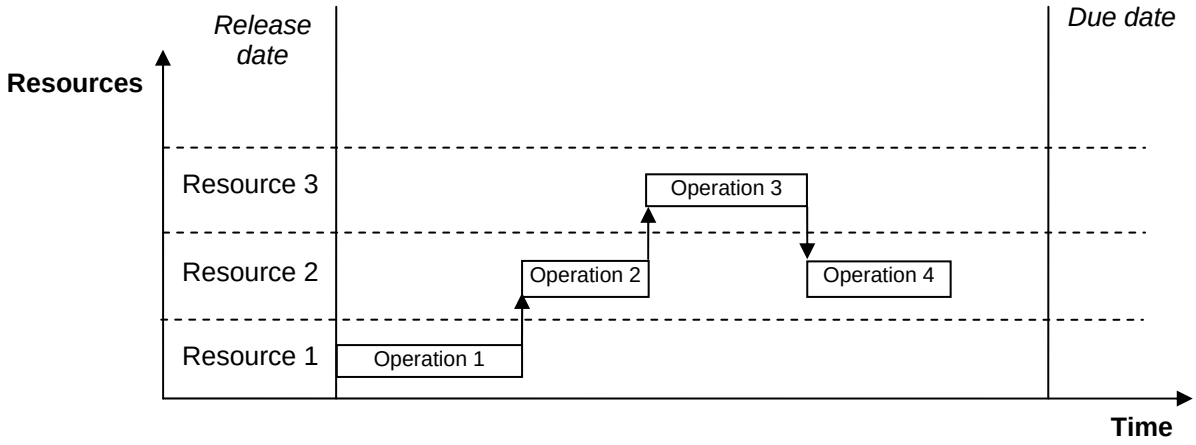


Figure 2.3. Forward scheduled order.

Backward scheduling – operations of an order are scheduled as late as possible backwards from the due date. See Figure 2.4.

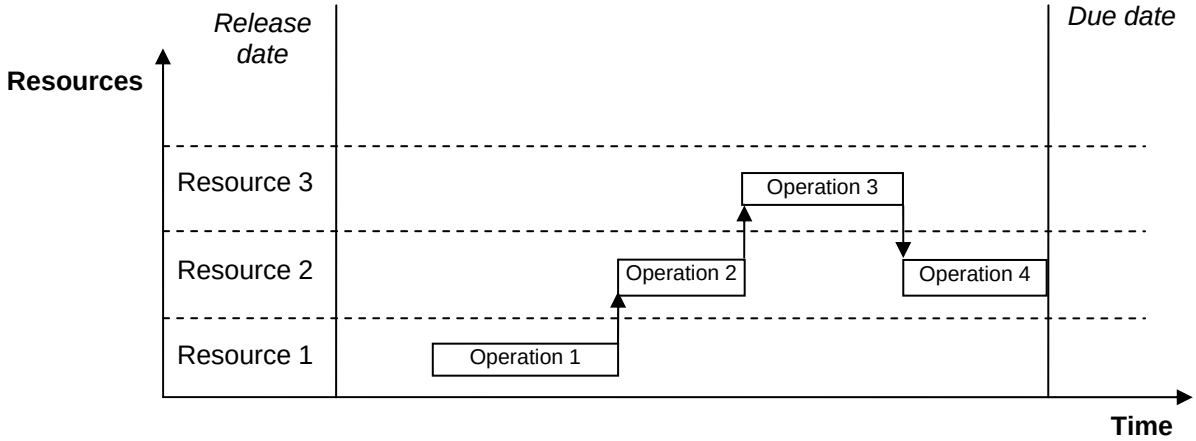


Figure 2.4. Backward scheduled order.

Static scheduling – the scheduling is performed on a fixed set of orders. This is the type of scheduling considered in this dissertation and also in much of the literature on scheduling. It can be viewed as scheduling a snapshot of a production environment.

Dynamic scheduling – new orders are continuously added during scheduling.

Production resources - personnel and/or machines that performs production in a company.

Work center – a production area consisting of production resources with similar capabilities.

Shift – a time interval describing when production resources are available.

Stock – stored products or parts ready for sale.

3 Production scheduling

As described in chapter 1, production scheduling consists of the activities performed in a company to determine in detail how work should be executed on the shop floor. How production scheduling is performed and the scheduling methods used, can vary a great deal between different production environments, from applying simple rules when choosing the next job to execute, to the use of advanced optimizing methods that try to maximize the performance of the given environment.

The scheduling approach suitable for an environment is highly dependent on how much complexity, uncertainty and randomness there is in the production system. This, together with the goals of the company, must be considered when choosing a scheduling approach for a particular environment.

Since scheduling involves prediction of the future, it of course gets more and more difficult when the level uncertainty and randomness in the system increases. A general rule is that as the uncertainty increases, the value of scheduling deceases [25]. Trying to predict and optimize the behavior of a complex system with many uncertainties and high level of randomness is in most cases a waste of time and resources. On the other hand, for more stable systems, putting more effort into scheduling often can improve performance and the use of advanced optimizing methods might in these cases be appropriate.

How scheduling is performed also depends on the management of the company and the organization around the scheduling function. Some companies may not have an explicit scheduling method; it is just something that is done implicitly by for instance the workers or shop floor management, while others have very strict approaches that are decided upon by the management.

How the schedule is used can also differ between companies and also between different users in the same company. Besides the obvious use on the shop floor other potential uses are to determine system capacity for a higher level production planning system, where the generated schedule is used to determine the feasibility of the production plan or by the sales department to determine if an order with a given lead time should be accepted [19].

3.1 Characteristics of production environments

Production environments can be characterized in a number of ways such as if the production is continuous (process industry) or job oriented (discrete manufacturing), if the products are made to stock or made to order [28]. In this dissertation the focus is on discrete manufacturing since continuous production involves other kinds of complexities that are outside the scope of this study.

In discrete manufacturing, shops are generally characterized as either flow shops or job shops. In a flow shop, orders are performed in a fixed sequence through the machines and other resources of the work centers in the shop whereas in a job shop, orders go through the work centers in arbitrary patterns. Scheduling in a job shop is therefore more complex than scheduling in a flow shop and the job shop is the more general case.

The type of demand that drives the production also affects scheduling. If the demands are known for a long time into the future, scheduling is easier than if demand is uncertain and changes must be handled on short notice. Production that is made to stock is often easier to schedule as the stock can be used as a buffer that gives freedom when scheduling. If the produced goods are delivered directly to customers, no such buffer exits since the customer expects that agreed upon delivery times are kept.

At a more detailed level, three components must be analyzed to determine viable scheduling approaches for the production environment. These are the production resources, the orders and operations, and materials and subparts.

3.1.1 Production resources

Production resources are everything that is required to perform the production. It can be for example personnel, machines or tools and other equipment. Resources with similar skills and capabilities are often grouped into work centers. Four characteristics can be used to describe resources: functionality, capacity, availability and cost.

3.1.1.1 Functionality

The functionality of a resource describes what operations it can perform. This is determined by for example the skills and competence of the personnel or the capability of the machine considered.

3.1.1.2 Capacity

The capacity of a resource can be described by how many jobs the resource can perform at the same time and by how effectively it performs an operation. It is possible that resources have

the same functionality but perform the same operation with different efficiency. For instance, a specialized resource can often perform an operation suitable for it faster than a general purpose resource.

3.1.1.3 Availability

This characteristic describes when the resource is available to perform operations. The availability of a resource to perform an operation is determined by when the resources is open (which shifts that are applied to the resource) and what other operations that are requiring the resource's capacity.

3.1.1.4 Cost

To perform production in a resource always incurs a cost. This cost must often be considered when scheduling. It is in most cases desirable to perform work where it incurs the least cost.

3.1.2 Orders and operations

Orders and operations describe what should be produced by a production environment and those activities that must be performed to accomplish this. The structure of orders and operations is hierarchical; an order contains operations and possibly also depends on other orders. Both orders and operations have costs associated with them. As the operations of an order are executed they accumulate cost that is derived from the costs of the production resources. The costs of the operations are then accumulated in the orders. Orders that are under execution are called work in process or in-process inventory. It is often desirable to have as few orders in process as possible and to keep the cost associated with those orders as low as possible.

3.1.2.1 Orders

Orders can be of different types such as customer orders which are associated with a specific delivery to a customer and/or a production order that can either satisfy the need of a customer order or be stored in stock for later use. Orders can have different states such as planned, in process or finished. Information associated with an order can be for instance release date, due date, quantity and/or priority.

3.1.2.2 Operations

An operation describes a basic activity or task that should be performed. The operations contained in an order have precedence relations between them that describe the sequence they should be performed in. The information associated with an operation varies with the

production environment, but some sort of description; which production resource or work center that should perform it and the expected processing time and quantity. These last two attributes are always required. This information can then be extended with information such as set-up time, post-production time and required tools. During order execution it is also necessary to keep track of the state of the operation such as not ready, started or finished. In more advanced production environments other possibilities might exist such as splitting an operation and performing it in parallel on more than one resource or having alternate resources that can perform an operation. It can also be possible to interrupt an operation to start another more important operation, this is called preemption. When the next operation should start can also vary, for instance after some specified quantity is produced or some specified time has elapsed on the current operation.

3.1.3 Materials and subparts

Materials and subparts that are required to perform production are specified per operation. Materials can be raw materials such as steel or wood or goods like nuts and bolts. Subparts are more refined components that can either be bought from a supplier or manufactured by the company and then stored for later use. To keep track of materials and subparts and to determine when operations can be performed information such as available quantity, expected deliveries and allocations to orders and operations must be maintained.

3.2 Scheduling objectives

At the most basic level, the reason for scheduling is to satisfy the overall goals of the company [4]. To be useful in the scheduling activity these goals are broken down into more detailed objectives. Some of the more common objectives are [24]:

- Meet due dates
- Minimize work-in-process inventory
- Minimize the average flow time through the system
- Provide for high machine/worker time utilization
- Provide for accurate job status information
- Reduce set-up times
- Minimize production and worker costs

An aspect that adds to the complexity of scheduling is that some of these objectives are in conflict with each other. Their origins are often in different departments of the company and

these departments have different goals. For instance meeting due dates is aimed at providing high customer service and is primarily a goal for the sales department whereas to minimize production cost and worker cost is primarily a goal for the production department. These two objectives can be in conflict if for example overtime is needed to keep due-dates. One important part of scheduling consists of balancing different objectives and to make decisions on how conflicts should be resolved.

3.3 Organization and information flow

How a company is organized around the scheduling function of course affects how scheduling is performed. Connected with this is whether the company has an explicit scheduling approach or not. Even if the company does not have an explicit approach some sort of scheduling occurs. In such cases scheduling is often performed implicitly by for instance the workers themselves or shop floor management by using higher level planning information to choose the next job to execute.

When an explicit approach exists, the scheduling can for instance be performed by a specific scheduling department or a scheduler that belongs to the production or sales department. The personnel performing this scheduling often have specialized skills and utilize more advanced methods.

Regardless of where in the organization scheduling is performed it involves a great deal of collaboration and information exchange. When a schedule is developed, demand information from higher level planning must be considered together with information on available resources and other constraining factors in the shop and information on available materials and sub parts from the material department. The schedule then has to be communicated to the shop floor where it is executed. The progress of execution also has to be fed back to the scheduler and other interested parties so that the outcome of the schedule can be analyzed.

3.4 Schedule evaluation and comparison

Determining the quality of a schedule and comparing different schedules is very important when choosing a scheduling method [19]. An optimal schedule performs the required production as effectively as possible and satisfies the overall goals of the company but in reality it is difficult to determine when this is the case or how far from being optimal a schedule is. A generalization of the most common goals that affects scheduling is to keep due dates and to minimize inventory levels [32]. Under these circumstances an optimal schedule is

one where every order is finished on its due date and each operation is performed as late as possible in the production resources with the lowest costs. This definition of an optimal schedule can be useful to have as a reference since the goals it satisfies are common in manufacturing companies.

How schedules should be compared in a particular production environment must be determined from case to case. To measure the performance of a schedule a set of metrics must be defined. The metrics must in some way be derived from the goals of the company. Examples of metrics are number of late orders (orders finished after due date), number of unavailable materials and subparts or number of resources that cannot provide needed capacity. The importance of different metrics and the relationships between metrics must be determined for the particular production environment in question.

3.5 Scheduling methods

The scheduling approach and methods that are suitable for a production environment depends, as described earlier in this chapter, on the characteristics of the environment, the complexity, uncertainties and randomness of the production system, the scheduling objectives and the organization around the scheduling function. Scheduling methods can range from simple rules for choosing which job to execute next, often called dispatching rules, to sophisticated optimizing methods. A very important general rule is [25]:

> *The more randomness there is in a system, the less advisable it is to employ very sophisticated optimization techniques. Equivalently, the more randomness the system is subject to, the simpler the scheduling rules should be.*

If advanced scheduling methods should be applied in an environment the uncertainties must as much as possible be reduced and the remaining uncertain factors must be handled in some way. The production environment must also be possible to represent as a model that can be used by a software system. This model must describe the properties of the environment in enough detail to make the developed schedules feasible to execute on the shop floor. The information in the model must also be updated to reflect the actual conditions in the production environment. To keep the model updated can require a lot of effort and this must also be considered when selecting which scheduling method is appropriate for an environment.

Another aspect is how long into the future the schedule spans, which is called the scheduling horizon. This is a continuation of the planning process described in chapter 2, it may not be useful to schedule the production in detail more than a limited time into the future, after that a more aggregated level is sufficient. What the scheduling horizon should be must be determined for the particular production environment in question.

The reason for using more advanced methods than just dispatching rules is to improve the performance of a production environment and to better satisfy the scheduling objectives. When using dispatching rules, the information used for choosing which job to execute is mostly local to the work center and this can result in sub-optimizations that do not contribute to the satisfaction of the scheduling objectives. More advanced methods tries to consider the global state of the production system to determine what actions should be taken to best satisfy these objectives. This bigger picture of course requires more information handling and when done manually relies primarily on the mind of the person doing the scheduling and the use of tools such as planning boards. It is here that a software system can be put to use to handle the information processing. To do this the information has to be formalized in some way into a model as described above. When using a software system for scheduling, the methods used can be much more complex than what is possible with a manual system.

In this section the two general groups of scheduling methods, deterministic and stochastic, are described in 3.5.1. Dispatching rules are overviewed in 3.5.2 and advanced scheduling methods in 3.5.3.

3.5.1 Deterministic and stochastic scheduling methods

The scheduling methods described in the literature can be grouped into the general categories deterministic or stochastic. In deterministic methods all variables of the model describing the problem are assumed to be known in advance and no uncertainty exits. Stochastic models have some, or all, variables defined as random and the methods use probability distribution when developing schedules. Stochastic scheduling methods used in practice usually involve some kind of dispatching rule.

Since a production system without any uncertainty does not exist, ways of handling uncertainty in deterministic models have been developed. Deterministic methods with some kind of support to handle uncertainty seem to be the most frequent in literature.

3.5.2 Dispatching rules

The most basic scheduling method is to use dispatching rules (also called priority sequencing rules) to determine which order to run next at a work center. These rules are applied when jobs arrive at a work center to choose the next task to be executed. Since dispatching rules only use information that is available at the moment when the next activity shall be selected, they work equally well in systems with a high degree of uncertainty as in more stable environments. When there are high levels of randomness and uncertainty in the production environment, dispatching rules may be the only viable way to schedule the production.

There exist many dispatching rules, some of the most common are [35]:

- *First come, first served (FCFS).* Jobs are process in the order they arrive at the work center.
- *Shortest processing time (SPT).* The job with the shortest processing time is processed first.
- *Earliest due date (EDD).* The job with the earliest due date is processed first.
- *Critical ratio (CR).* A priority index is calculated using (time remaining / work remaining). A ratio less than 1 means that the job is late. The job with the lowest ratio is processed first.
- *Least work remaining (LWR).* Priority based on all processing time remaining until job is completed.
- *Fewest operations remaining (FOR).* Priority based on number of remaining operations.
- *Slack time (ST).* Jobs run in the order of the smallest amount of slack.
- *Slack time per operation (ST/O).* Slack time is divided by the number of remaining operations. Jobs are sequenced in order of smallest value.
- *Next queue (NQ).* The queues in front of successive work centers are measured (in hours or number of jobs). The job that is going to the smallest queue is processed first.
- *Least setup (LS).* The job with the least setup time is processed first.

The general properties of these rules are different [35]. SPT, and its variations LWR and FOR, reduces work in process inventory, average job completion time and average job lateness but can cause starvation of jobs with long processing times and thus cause missed due date. EDD, and its variations ST and ST/O, reduce job lateness but result in higher average time in the system. NQ and LSU maximize machine utilization. There exist many other dispatching rules and also variations of the above rules. To combine rules, for instance using different rules for different work centers is also possible.

Scheduling using these rules can, depending on the scheduling problem, give good results but there is a risk of sub-optimization since the information used is local and no consideration is given to the global state of the production system.

3.5.3 Advanced scheduling methods

Advanced scheduling methods use more information when developing schedules than dispatching rules and try to consider more or less of the global state of the production system. In industry these methods are categorized as Advanced Planning and Scheduling (APS) methods. The APS category includes methods for scheduling as well as demand management, production planning, distribution planning and transportation planning [10].

Advanced scheduling methods require that the production system is represented as a model that can be used as a good enough approximation of reality to make the developed schedules executable on the shop floor. The model must describe all the relevant constraints in the production system and include information such as:

- Orders, operations and precedence relations between these
- Resources required to perform the operations and their available capacity
- Materials required to perform the operations and the availability of these
- Release and due dates of orders
- Priorities among orders
- Real and expected costs incurred by the different activities and decisions

What is a good enough approximation must be determined from case to case, but there should always be a correspondence between how well the model describes reality and how advanced the used scheduling method is. The general rule, the more uncertainty the simpler scheduling method, applies here. If the developed schedule is not feasible to execute on the shop floor because of missing or faulty information in the model, applying advanced optimizing methods is a waste of time and resources. Effort should in those cases instead be put into developing and refining the model in combination with measures to remove uncertainties from the production system.

3.5.3.1 Finite capacity scheduling

In industry, applying advanced scheduling methods is often called finite capacity scheduling (FCS). Traditional methods such as MRP consider the capacity of production resources more or less as infinite. When the constraints imposed by the capacity actually available in the

production resources are considered by the scheduling method this is termed FCS. Scheduling with and without consideration of capacity constraints is illustrated in figures 3.1 and 3.2.

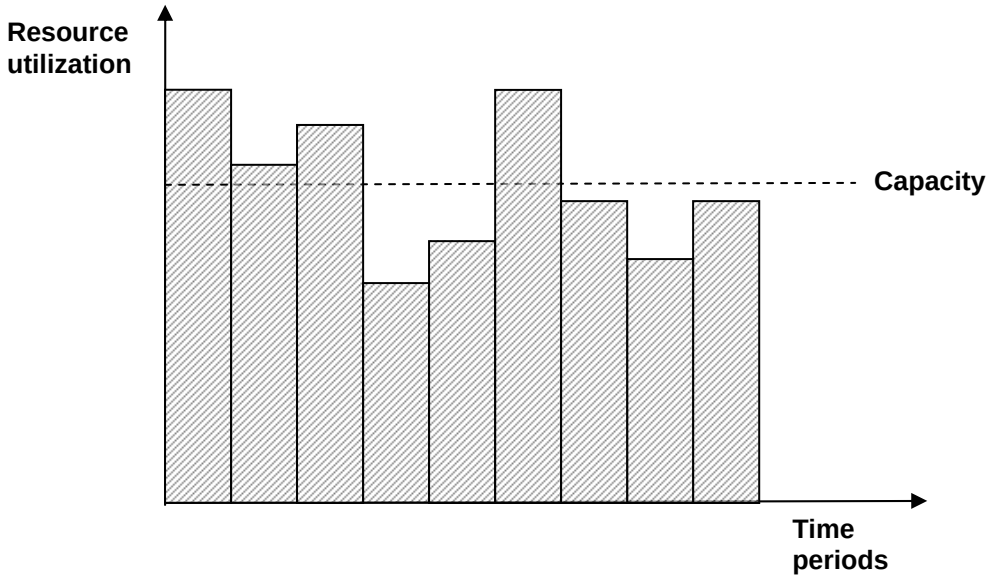


Figure 3.1. Scheduling allowing infinite capacity load.

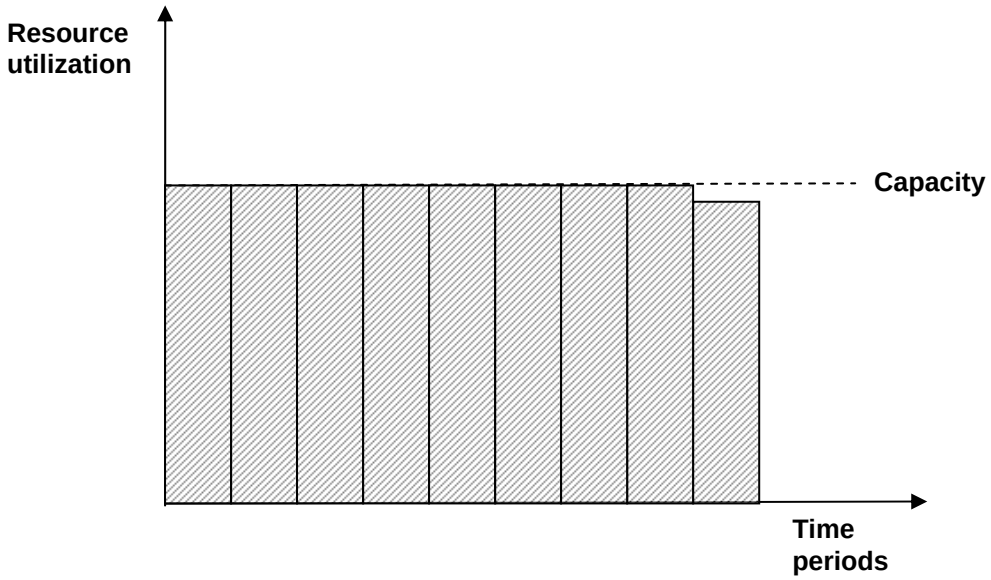


Figure 3.2. Scheduling with finite capacity load.

In FCS jobs are only scheduled on a resource up to the resource's capacity limit. This means that jobs that cannot be performed at the required time are going to be moved forward, possibly making jobs late. FCS makes it possible to find those resources where available capacity is not enough to perform the required work load. Such resources are called bottleneck resources and finding these problem spots is very important when making schedules. Just that a schedule is considering the constraint imposed by the finite capacity of the production environment says nothing of how well it satisfies the goals of the company. What can be said is however that the more advanced methods all must be developed against finite capacity to be feasible and possible to execute on the shop floor. A basic example of FCS can be a method that applies dispatching rules to all orders planned to be executed over some time period in a production environment while at the same time considering the constraints imposed by available capacity.

3.5.3.2 Considering availability of capacity and material together

Most advanced scheduling methods consider available capacity as finite, and thus can be termed FCS methods. Since production activities often require some kind of material or sub-parts to be performed, the availability of those must also be considered if feasible schedules are to be developed. If materials and/or sub-parts are not available, the operation cannot be performed and must be moved forward to a time when the material/sub-part becomes available.

A problem in MRP is the separation between material and capacity planning (see section 2.2.3). This is solved in advanced scheduling methods by considering capacity and material constraints at together when developing the schedule.

3.5.3.3 Characterization of advanced scheduling methods

Advanced scheduling methods can be characterized by the way they decompose the problem [26][25]. Decomposition approaches to scheduling problems can be grouped into the three categories job-based, resource-based and event-based. These approaches can also be combined in different ways to suit different production environments.

Job- based methods

In job based methods scheduling is performed at the job, or order, level. In [35] this is referred to as horizontal loading. All jobs that are to be scheduled are given a priority. The priorities can be based on different factors such as due dates or other information determining the importance of jobs. The schedule is then developed by scheduling the jobs one at time in priority order, from highest priority to lowest. All operations that belong to a job are scheduled at the same time and required resource capacity and materials are allocated for these operations ahead of time. Job-based methods result in schedules that get jobs finished in priority order. Maintaining a high level of resource utilization is not considered. An effect can be that a work center is idle waiting for a high priority job to arrive while other, lower priority, jobs are ready to execute.

Resource-based methods

Resource-based methods decompose the problem by the utilization level of resources and places focus on bottlenecks in the production system. The methods are therefore often referred to as bottleneck methods. The bottleneck method that first got attention was developed by Eli Goldratt who has termed it the theory of constraint (TOC) [15][6]. The basic idea of TOC is that the constraints of a system determine its overall performance. A constraint according to Goldratt is "*anything that limits the performance of a system relative to its goals*" and the method actually can be applied to more than just scheduling [6]. In the scheduling part of the theory, the method is to schedule the bottleneck resources very rigorously and use simple methods to schedule all other, non bottleneck, resources. The bottleneck resource determines the throughput of the whole system and therefore its utilization should be maximized. A buffer is put before the bottleneck resource to ensure that it never has to wait for jobs to execute. The concept is called drum-buffer-rope where the bottleneck resource is the drum that determines the pace, the rope is the signaling mechanism that links the other resources to the bottleneck and the buffer insulates the bottleneck from the rest of the system. The method's purpose is to maximize the throughput of the system. By focusing on the bottleneck resource and letting the other resources follow, more effort can be put into scheduling just the problem resource.

TOC requires that there exist just a few bottleneck resources and that they be well defined. In reality this is often not the case, it is common that the bottleneck resource shifts over time. In [1], a procedure for scheduling such cases is described.

Event-based methods

Event-based methods decompose the problem at the operation level. These methods determine which task to be executed in a resource at the operation level. Instead of scheduling all operations included in a job at the same time as in job-based methods, the operations are scheduled separately. The precedence relations between operations in a job must of course be maintained while doing this. In [35] this type of scheduling is referred to as vertical loading.

In event-based methods time is considered to be moving forward event by event. The events considered are occurrences that change the state of the system and require some action to be taken. Such events are for example, operations becoming available for execution at a work center, operations being finished at a work center (making resource capacity available) and material becoming available for use by an operation. At each event the current state of the system is considered and actions are taken, such as starting a new operation in a work center. By decomposing the problem at the operation level, potentially more possibilities are available to apply rules that prioritize operations according to different weighted objectives. An example of such a rule can be to schedule the operations in job priority order but if there is available capacity at a work center and current demand exists for that capacity by operations that are not in priority order, then execute those operations even if this makes higher prioritized operations late. An example of an event-based method is the micro-opportunistic scheduling described in [32].

3.5.3.4 Knowledge based and decision support systems

Another approach that deserves mention uses knowledge based or decision support systems to perform scheduling. Knowledge-based systems model the production environment and use knowledge elicited from existing scheduling experience to find feasible schedules. These types of system are also called expert systems. The elicited knowledge is formalized as rules that are applied to the model.

3.6 Scheduling in practice

Despite the potential usefulness of a supporting software tool to aid the scheduling activity, for instance to process large amounts of information, facilitate communication and ultimately produce better schedules, the implementation and use of scheduling software in practice has been found to pose a number of problems [4][22][29][33]. These problems can be grouped into two basic categories, one that encompasses problems encountered when modeling the production environment and how to handle the uncertainties that will always exist, and the other consisting of problems related to human factors when it comes to using the systems.

3.6.1 Practical modeling and handling of uncertainties

As related in section 3.5.3, most production environments are highly dynamic and include many constraints that have to be described in sufficient detail to produce a model that can be used for scheduling. This model is an approximation and the production environment always contains some degree of uncertainty that is not included in the model but still has to be handled in some way. Some of the more common causes for uncertainties are [33]:

- Actual processing times differ from planned (often the actual time is longer)
- Actual capacity differs from expected (machine breakdowns, absence of personnel)
- Changing priorities (rush orders)
- Insufficient feedback from production (performed work not reported)

When scheduling manually, uncertainties are handled by the experience and intuition of the scheduler. Manual scheduling also often use dispatching rules which are not affected by disturbances in the same way as a more advanced method. Every time a dispatching rule is applied, only the current state of the system is considered whereas advanced methods often try to consider future events which make them more sensitive to disturbances.

How well a schedule handles uncertainties is termed the robustness of the schedule. The robustness of a schedule is a measure of how much disturbance that can occur in the production environment before the scheduling has to be redone. Robustness can be achieved for example by inserting slack into the schedule and/or by not utilizing production resources over a certain level (e.g. 80% of actual available capacity). Such measures insert buffer time into the schedule that can be used to handle unforeseen events.

When to perform rescheduling is something that has to be decided for each particular environment. Rescheduling can for instance be performed on a daily basis, when new orders

are released for production or when the disturbances have reached such a level that a new schedule is necessary.

A fact that further complicates this issue is that the data quality in most companies' information systems is often not sufficient. The information is in many cases faulty, missing or inconsistent. Since advanced scheduling methods require high quality data to produce feasible schedules it is often the case that activities to increase data quality is needed before it is possible to apply such methods.

3.6.2 Human factors

Traditionally scheduling has been done manually and the persons responsible for scheduling normally have a thorough knowledge of the production environment and use this together with skill and intuition to develop the schedules. There is often much personal contact between the scheduler and other interested parties such as production personnel and sales department and as stated in [16], scheduling in reality is very much a social activity. This is especially important when the environment contains uncertainty and randomness since personal contact and the information exchanged through those contacts are crucial when handling such circumstances. When implementing a scheduling tool it is very important that the human aspects are considered. The software system must be felt to be an aid in the scheduling activities and adapted to the particular production environment where it is used. Experience has shown that many implementations have failed because the tool does not fit the environment and because the interested parties do not have confidence in the schedules produced by the system.

It is apparent that a scheduling tool must be suited to the environment and that the scheduler and other interested parties must feel confident in the schedule produced by the system. It is therefore important that how the schedule is developed is transparent and possible to understand. An aspect to consider is also that the more advanced and complicated the scheduling method gets, the more skill and effort is also required by the scheduler. When determining a suitable scheduling method for a production environment this must be considered.

It is also necessary to have reports and other graphical representations that visualize the information used when developing a schedule. The visualizations should support the human abilities such as to recognize patterns in data, handle unexpected events and inexact information as described in [16].

The developed schedule must also be communicated to other interested parties. For instance, making the schedule understood and followed on the shop floor is an absolute necessity if a scheduling tool is to be useful and getting feedback from the shop floor is essential when the produced schedule should be analyzed. Good reports and other visualization tools are an aid in all these activities.

4 Scheduling theory

Despite the apparent difficulties with applying scheduling theories in practice it is of course important to investigate the available research to gain further insight into the problem and find out which solution methods can potentially be applied in reality. As mentioned before a large amount of research exists in the area of scheduling in different disciplines such as industrial engineering, operational research, artificial intelligence. In addition to this, there also exists research on production scheduling in disciplines such as psychology and decision science. The fact that research is spread across different research communities is actually a problem since it makes it hard to gain an overview of available research [36]. Much of the information in this chapter is derived from [5][18][36], which are papers that provide overviews on the subject.

Production scheduling research as related here goes back at least to the 1950's and still is a very active area of interest with new solution methods being developed continuously. The focus seems to have shifted from operational research to artificial intelligence when it comes to finding new methods [18][5].

The purpose of this chapter is to give a brief overview of the theoretical problem formulation and some of the solution methods that can potentially be used in practice. It is outside the scope of this dissertation to go into the details of the different algorithms and theories. Since the research area is very broad and spread across different disciplines it also often requires deep background knowledge from diverse areas. In a chapter like this it is only possible to present a summary and try to find the areas that seem most relevant when it comes to practical usage.

4.1 The scheduling problem

Scheduling can be defined as the problem of assigning scarce resources to competing activities over a given period of time [19]. The problem can be characterized as a combinatorial optimization problem and it has been shown to be NP-hard[2] and also to belong

[2] The term NP-hard referes to the theory of NP-completeness. This theory is used to classify problems with regards to how difficult they are to solve. A problem is NP-complete if there exists no algorithm that solves the problem in polynomial time. In polynomial time means that the time required by the algorithm to solve the problem grows polynomially with respect to the size of the problem. A problem is NP-hard if it can be shown to be at least as hard to solve as a NP-complete problem [14].

to the most intractable problems considered [5]. What make this problem so difficult are the same reasons which make many other real world problems difficult [23]:

- The number of possible solutions is so large that an exhaustive search to find the best answer requires far too much time
- The problem is so complicated that just to facilitate any answer at all, the model has to be simplified to a degree that makes the derived solution difficult to apply in practice
- The possible solutions are so heavily constrained that constructing even one feasible answer is difficult, let alone searching for an optimum solution

4.1.1 A problem formulation

There exists different problem formulations as related in [5] but a common representation that was developed in the 1960's seems to be the most used [1][5][32][18]. In this formulation the problem is modeled as a disjunctive graph. The graph representation gives a good visual presentation of the problem.

The basic model consists of a set of orders (or jobs) $\{J_1, \ldots, J_n\}$ which have to be scheduled on a set of resources $\{R_1, .., R_m\}$. Each job J_i consists of a set of operations $\{O^i_{1, \ldots,} O^i_p\}$ where $(1 <= i <= n)$. The operations are partially ordered, the most common case in practice is that an operation can have several predecessors but at most one successor. The operations order is described by their lower indices (O^i_k must be executed before O^i_{k+1}). Each operation has an expected duration and a start time (to be determined).

In the graph each operation is represented by a vertex and precedence relations between operations as directed edges between vertices. There are also two dummy vertices, 0 and n, representing the start and end of the schedule. Each vertex representing an operation contains information on the resource required to perform the operation and the operations duration. Each pair of operations that require the same resource is connected with an undirected edge. These edges represent capacity constraints. As an alternative, the undirected edges representing capacity constraints can be represented as a pair of directed edges. In some cases the duration is represented as the weights of both the edges representing precedence constraints and the edges representing capacity constraints [1][5]. Figure 4.1 shows an example adapted from [32] of such a graph for a scheduling problem with twelve operations belonging to four orders. The operations should be executed in five resources, $R_1 - R_5$. The

first operation of order 1 should be performed in resource 1 and have a duration of 2. In Figure 4.1, the capacity constraints are represented as undirected edges.

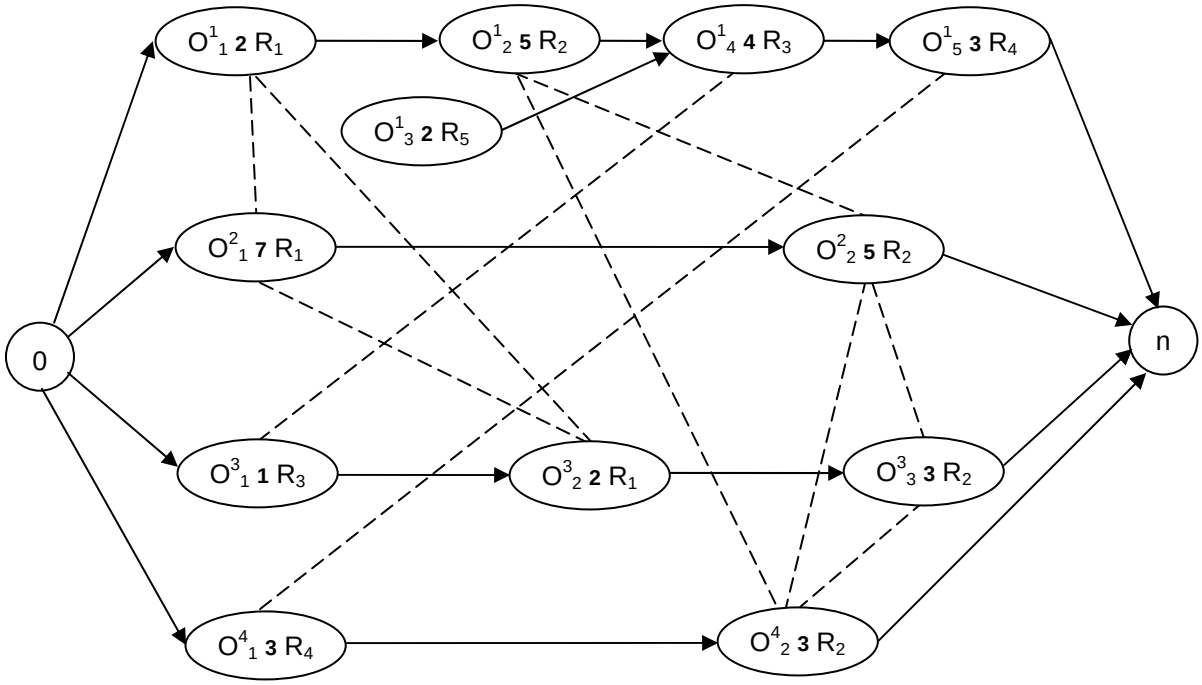


Figure 4.1. A graph representation of a scheduling problem.

Depending on the scheduling objectives the problem can be represented in different ways. For instance, if the objective is to minimize the total time it takes to perform all operations in the graph, the problem is to find the direction of the edges representing capacity constraints that minimizes the length of the longest path from 0 to n.

4.2 Solution methods

4.2.1 Problem solving

The process of problem solving consists of the two separate general steps, first to create the model of the problem and then to use the model to solve the problem. Every solution is derived from the model and how well the solution solves the actual problem depends on how well the model describes the problem. The difficulties in modeling production environments are described in section 3.6.1. Every algorithmic approach to problem solving have three concepts in common that has to be specified, the representation, the objective and the

evaluation function. The representation encodes alternative candidate solutions for manipulation, the objective describes the purpose to be fulfilled and the evaluation function returns a value that indicates the quality of the solutions [23].

The representation

A scheduling problem representation has to include orders, operations and precedence relations between these, resources required to perform the operations and their available capacity, materials required to perform the operations and the availability of these, release and due dates of orders, priorities among orders and real and expected costs incurred by the different activities and decisions. (See section 3.5.3). A possible representation is described in section 4.1.1. The difficulty lies in representing the complexities and constraints of a production environment in enough detail so that the solution, in this case the schedule, derived when applying the solution method is useful while still keeping the representation as simple as possible.

The objective

The objectives to be fulfilled when scheduling, are at the highest level the overall goals of the company, as described in section 3.2. These goals have to be in some way reflected in the algorithmic approach used to solve the problem. As described in section 3.2, the goals can often be in conflict in each other. These conflicts must be either resolved before the algorithm is applied or handled in the algorithm, for instance by putting weights on the used goals.

The evaluation function

As described in section 3.4, evaluating schedules can be difficult. The metrics that are used to evaluate the developed schedule must be chosen so that they reflect the quality of the schedule in a way that that is appropriate for the algorithm used. The evaluation function is also called the accept–reject criterion.

4.2.1.1 Exact and approximation methods

The space of all possible solutions to a problem is called the search space of the problem. Among the possible solutions in the search space, the only solutions of interest are those that are feasible. Feasible solutions satisfy the constraints imposed on the problem. Constraints

can be grouped into hard and soft, hard constraints are those that must be satisfied if the solution is to be feasible and soft constraints are those that are preferred to be satisfied but can be relaxed to find a solution. In scheduling, hard constraints are for example precedence relations between operations and the availability of materials and soft constraints are for example due dates that need to be relaxed because of missing materials or resource capacity that has to be increased by working additional shifts or overtime. The search space for a realistic scheduling problem with its many variables and constraints of course may get very large. Because of this, for most scheduling situations there exists no efficient algorithms that solves the problem. An efficient algorithm solves a given problem optimally with a requirement that increases polynomially with respect to the size of the input. In operational research, different applications of exact solution methods such as mathematical programming procedures, like linear programming and dynamic programming, have been applied, but none of these have successfully solved problems of the large sizes common in practice. It has been shown that efficient algorithms cannot be found for scheduling problems that encompass more than three operations and three jobs [18]. In reality, many production environments work with hundreds of orders and thousands of operations so other solution methods are required.

The types of solution methods that can be applied to scheduling problems are approximation procedures. Approximation methods do not guarantee achieving exact solutions but are able to attain near optimal solutions within moderate computing times [18]. The current trend seems to be methods coming from artificial intelligence like heuristic search methods, evolutionary algorithms and constraint programming.

4.2.2 Dispatching rules

The dispatching rules described in section 3.5.2 are approximation methods. Much research has been put into dispatching rules since they are easy to implement and require low computational effort. The research includes studies on the relative performance of the different rules in different production environments and how the rules can be combined [18]. Dispatching rules are also often used in other more advanced methods to for example generate a first solution that is then refined, or to solve smaller decomposed parts of the problem during the application of an algorithm.

4.2.3 Bottleneck based methods

Bottleneck methods decompose the search space at the resource level as related in section 3.5.3.3. Probably because the Theory of Constraint as developed by Goldratt (see section 3.5.3.3) is a proprietary method, it is not mentioned in the papers overviewing scheduling theory studied for this dissertation. Instead it is the shifting bottleneck procedure (SBP) described in [1] that is most often refered to. Since TOC requires one resource to be the bottleneck, SBP can possibly be seen as a refinement of TOC.

SBP works by decomposing the problem into a series of one resource (or machine) problems and by that subdividing the search space. Described briefly the procedure works as follows. The resources are scheduled successively one by one and the resource chosen to be scheduled is the one identified as the bottleneck resource among those not scheduled. Every time after a new resource is scheduled the previously scheduled resources are locally reoptimized. In SBP the problem representation described in section 4.1.1 is used.

SBP been further developed and refined by others as related in [18]. According to [34], the performance of bottleneck methods compared to using other methods is good if the difference in capacity between bottleneck and non bottleneck resources is significant.

4.2.4 Local search methods

Since exhaustingly checking each and every solution in a search space as large as those associated with scheduling problems is impossible, search methods that find feasible solutions without an exhaustive search are required. A way to avoid having to search the entire search space is to use heuristics, or knowledge of the problem domain, to subdivide the search space and to guide the search for feasible solutions. Heuristic search methods can also be termed problem space based methods.

Traditional methods that use heuristic guided local search are iterated hill-climbing methods and it is useful with a brief description of how these works since they are the basis of how more advanced methods work.

Hill-climbing methods

Hill-climbing methods use an iterative improvement technique that from a chosen current point in the search space tries to find a point in the neighborhood that provides a better value in terms of the evaluation function. Neighborhood in this case is intuitively defined as all points in the search space that are close in some measurable sense from the current point. The points in the neighborhood are iteratively searched until no improvement is possible or some other condition is met. Such a method can only find a local optimum that depends on the start

point. The starting point can be chosen at random or by some heuristic and to try to escape from the local optimum many different starting points can be chosen and the results compared to find the best solution. Despite starting from different points in the search space it is often very difficult to find a global optimum with these kinds of methods [23].

In scheduling, local search methods can be termed improvement techniques as they start with an existing schedule and try to improve it. The initial schedule can for example be developed by using dispatching rules or by some basic finite capacity scheduling procedure. Local search procedures for scheduling can be compared by the following four criteria [25]:

- The schedule representation needed for the procedure
- The neighborhood design
- The search process within the neighborhood
- The acceptance – rejection criterion (also referred to as the evaluation function)

The schedule representation can for example be a data structure holding all operations with information on start and end times and assigned resources. Two schedules can be defined as neighbors if one can be obtained through a well defined modification of the other. The design of the neighborhood is a very important aspect of a local search procedure. For most scheduling problems that exist in reality the neighborhood of a schedule is often complex and different methods have been developed to determine how these can be designed [25][18]. The search method determines how the next schedule to be evaluated is chosen and the acceptance – rejection criteria gives the value by which the schedule is compared.

Three local search algorithms that have shown to be suitable for practical use are simulated annealing, tabu search and genetic algorithms [25].

4.2.4.1 Simulated annealing

Simulated annealing is a search process that has its origin in the fields of material science and physics and was first developed as a simulation model for describing the physical annealing process of condensed matter. The accept–reject function in simulated annealing is based on a probabilistic process, when a feasible solution is found during the search procedure it is either accepted or rejected depending on a stochastic variable called temperature. The accepted solution need not to be better than the current solution, it is sufficient that it is feasible. The reason for allowing such moves is to give the procedure a chance to move away from a local optimum and find better solutions later. The temperature variable used in the accept–reject

function is updated as the procedure progresses so that the probability that non-improving solution are chosen decreases. The stopping criteria for the process can be for example a predetermined number of iterations or a predetermined number of iterations without improvement.

4.2.4.2 Tabu search

Tabu search is similar to simulated annealing in that it moves from one schedule to another with the next schedule being possibly worse than the previous. The basic difference between the two methods lies in the way a new schedule is accepted or rejected. In tabu search the process is deterministic rather than probabilistic, and is based on the history of the search. During the process a list of moves that is not allowed is kept, called the tabu-list. The idea is that the tabu-list serves as a memory that should force the process to search in new areas of the search space. Solutions that have been examined recently are stored in the tabu-list and are not considered when searching for the next solution, they become tabu. Like in simulated annealing the idea is to give the procedure a chance to escape local optimums and find better solutions in other areas of the search space.

4.2.4.3 Genetic algorithms

Genetic algorithms are modeled after the theory of evolution in which the fitness of an individual determines its ability to survive and reproduce. When genetic algorithms are applied to scheduling, schedules are viewed as individuals in, or members of, a population. This is one aspect that differs from simulated annealing and tabu search, in those algorithms only one schedule is carried over to the next step whereas genetic algorithms work with a population consisting of a number of schedules that are transferred between iterations. Iterations are called generations in genetic algorithms. The population size often remains constant between each generation. Each individual in a population is assigned a fitness value that is derived from the evaluation function. The schedules to be carried over from one generation to the next can either be transferred as is if their fitness value are good enough, or generated through reproduction and mutation of schedules in the previous generation. Schedules with low fitness values are disregarded. The reproduction and mutation procedures use heuristics from the problem domain and probability to mimic the corresponding processes that occur in nature. The idea is to arrive at a schedule that is near optimal by letting the population of schedules undergo the reproduction and mutation transformations from

generation to generation. How genetic algorithms can be applied to practical scheduling problems is described in [27].

4.2.5 Constraint programming

Scheduling problems can be characterized as constraint satisfaction problems (CSP's) [21][12]. A CSP is defined by a set of variables V = $\{v_1, \ldots, v_n\}$, each having a domain D = $\{d_1, \ldots, d_n\}$ and a set of constraints C = $\{c_1, \ldots, c_n\}$. The domain of a variable can have many different characteristics such as an interval of integers or Boolean true/false values. Two domains that occur frequently in scheduling problems are the temporal domain which is infinite and different finite domains which consist of a finite set of values that for example can describe production resources. A constraint is a m-tuple that specifies a consistent assignment to the variables that it constrains, $c_i \subseteq d_1 \times d_2 ... \times d_m$. The process of solving a CSP basically consists of three steps:

1. Select a variable for instantiation.
2. Select a value from the variables domain and assign this to the variable.
3. Determine if the assignment is consistent with all constraints defined for the problems. If the assignment is not consistent with the constraints backtrack, that is go back to step 1 and either select another value for the variable or select another variable to assign a value to. If the assignment is consistent with the constraints go to step 1 and select a new variable.

The process ends when a specified number of solutions is found in which all variables are assign values and these assignments are consistent with the constraints or it is determined that no solution exists. How effective this process is depends on how the variable to assign a value is selected and then what value it is assigned. The goal is to make selections that as fast as possible minimizes the search space by for each iteration of the procedure be able to remove as many infeasible values from the domains as possible while at the same time finding a solution with a minimal amount of backtracking. Much research has been put into finding heuristics for how this selection process should be performed.

Constraint programming is a programming technique that is used to solve CSP's. Constraint programming was first an extension of logic programming languages, appropriately called constraint logic programming (CLP), but there now exist constraint programming libraries for other languages as well. To solve scheduling problems it is

common to use constraint programming toolkits together with object-oriented languages. This gives the possibilities to model the problem using object-oriented design methods and then solve it using constraint programming.

When using CLP languages or constraint programming extension for other languages the idea is to state the problem at a high level by specifying variables with their domains and the constraints that must be maintained, and then leave the solving to mechanisms build into the language or libraries. The mechanisms that perform this are called solvers and much research is put into the development and design of solvers for different applications.

Examples of practical use of constraint programming to solve scheduling problems can be found in [32][20][8].

4.3 Summary

As mentioned in the beginning of this chapter it is only possible to present an overview of the large area of scheduling theory and the chapter should be seen as an orientation. The methods and algorithms mentioned in the chapter all seem possible to apply in practice and the listing can serve as a starting point when searching for methods to tackle scheduling problems that occur in real production environments. In practice the methods can also be combined in different ways, for example using combinations of dispatching rules to develop an initial schedule that is then further improved by a local search method such as simulated annealing or tabu search [25].

To actually use and implement the methods it is of course necessary to study them more thoroughly. Sources of information for further studies can be found in the references used for this dissertation. The references also contain many other methods and algorithms that are not mentioned in this chapter. Some of these, like neural networks and distributed agent based scheduling methods, seem to attract a lot of interest in the research communities and may provide alternative solution methods in the future.

5 Software systems for manufacturing planning and control

This chapter gives a brief overview of the different software systems used to support MPC and production scheduling.

5.1 Enterprise Resource Planning systems

The software used for MRP often consists of large suites of integrated programs designed to support most of the activities in MPC plus accounting and other functionality. These systems are called Enterprise Resource Planning systems (ERP-systems). The systems are usually built upon a relational database and are deployed in a distributed client-server environment. The information used for scheduling purposes is for the most part maintained in the database of the ERP-system. What this information consists of is described in sections 3.1 and 3.5.3. Some ERP-vendors also integrate third-party scheduling components into their systems or provide an interface to some external scheduling program.

5.2 Scheduling systems

There exists a lot of different scheduling software or Advanced Planning and Scheduling (APS) -systems as they are often called in industry. The scheduling system can either be a complete free standing system or a scheduling component that is integrated with the ERP-system.

5.2.1 Complete systems

The basic architecture of a scheduling system consists of the following modules [25][37]:

- Database access module to communicate with the ERP-system and other data sources to retrieve and update scheduling data.
- A scheduling module to generate and manipulate schedules. This is also called the scheduling engine. This part of the systems contains the scheduling algorithms.
- A user interface module which allows the user to interact with the system.

5.2.1.1 Database access and communication

Database access is most commonly performed via SQL and the standard interfaces that are available for this. This part of the architecture is probably the least complicated and should be straightforward to implement. Most data sources support SQL and via this it is possible to have flexible interaction with different data sources.

5.2.1.2 Scheduling module

In the scheduling module the data must in some way be transformed into a representation that is appropriate to use when applying the scheduling algorithm. To do this the module must have flexible modeling capabilities to be adaptable to different production environments. The algorithms used in the module are often possible to extend and modify by the user. It is common that that some simpler algorithms are provided, such as basic finite capacity algorithms using dispatching rules, and that these then can be modified to suit the particular environment.

5.2.1.3 User interface

The user interface part of the system is very important since it is often this that determines if the system is going to be used or not. The user interface should be as simple and intuitive as possible so that the user can concentrate on the scheduling task. Some types of interfaces that are typically available in a scheduling system are an interactive Gantt-chart to view and manipulate the schedule, graphical representation of load profiles like those in Figure 3.1 and Figure 3.2 (possibly with possibilities to manipulate resource capacity) and interactive dispatch lists. There must also be interfaces for controlling loading and updating data to and from the ERP-system and other data sources as well as for controlling scheduling parameters.

5.2.2 Scheduling components

Scheduling components contain the functionality available in the scheduling module described in section 5.2.1.2. The component can be integrated into ERP-systems and can also be used by scheduling system providers that do not want to implement this module themselves. Using a scheduling component can offer more flexibility since it should be possible to integrate different kind of components depending on the scheduling requirements. One common type of scheduling component is constraint programming based solvers that is possible to integrate into object oriented frameworks [20][8].

5.3 Manufacturing execution systems

Manufacturing execution system (MES) is a category of system that is intended to be used on the shop floor. They include for example process controllers and other systems to control machines and equipment but from a scheduling perspective what is important is the possibility to communicate and get feedback on the schedule using these systems.

6 A Case-Study

To obtain some experience of the requirements that might exist on a scheduling system in reality, a case study has been performed. The study has been done in a job-shop that maintains and repairs electrical equipment (motors, transformers etc.) where a finite capacity scheduling system is in use. The scheduling system is free standing program that is integrated with the company's ERP-system. By performing the study in an environment where a scheduling system is already in use, the requirements come from the user's experience of using such a system in their daily work and also the framework around the system is in place and can be used as a base when testing and evaluating the developed prototype. The current scheduling system has been in use for about 3 years and has solved some of the shop's scheduling problems but it is felt that there are still a number of improvements that can be made.

The purpose of the study is not to evaluate the current system but to investigate what requirements there are on a scheduling system through all steps in the production process in this shop. Of course, to some extent, the user's views are based on the features, and also the shortcomings, of the current system but the aim has been to focus on the general scheduling process.

6.1 Production environment

6.1.1 General description

The job-shop studied maintains and repairs electrical equipment for a number of different customers. The customers send their equipment to the shop where it is maintained and/or repaired and then sent back. Since the equipment is often critical for the customer, keeping due-dates and having short lead-times are very important. Maintenance and repair differs from manufacturing of new products in that the production is much harder to predict, maintenance can to some extent be planned in advance but equipment which breaks down and needs repair cannot. The scope of work of every object is also more or less unique and hard to define in advance and lead-times of orders can vary from a couple of days to several months. It is also quite common that additional work on an object is discovered during job execution which means that the original scope of work is changed. This of course affects scheduling in that uncertainty and unpredictability must be handled to a greater extent than perhaps is usual and,

since each object is unique, no batches of objects on the same order are considered. Normally there are between 150 – 200 objects with about 2500 operations in process so to manually schedule the shop in detail is in practice impossible. Some kind of scheduling support system is therefore needed and by implementing the current scheduling system the goal was to improve the scheduling in the shop by:

- Obtain a global view of the current production situation that is shared among all participants in the scheduling process.
- Take informed scheduling decisions that satisfy the overall objectives of the shop and consequently minimize local, potentially sub-optimizing, decisions.
- Execute, and provide feedback on, the schedule decided on the shop floor
- Reduce lead-time and improve the accuracy in the due dates that are promised to customers
- Solve problems early, hopefully before they disturb the production and affects deliveries.

6.1.2 The job shop and order structure

Work centers and resources

The job-shop consists of 15 work-centers that handle different types of tasks and each work center consists of a number of resources. The resources are all production personnel, no machines are scheduled since it is the operator using it that is the constraining factor. To handle uncertainties, the possible utilization rate of most work centers is set to less than 100%. This means for instance that if the utilization rate of a work center is set to 80%, and if the planned duration of an operation is planed to be 1 hour, it is actually scheduled to take 1.2 hours to perform in this work center. This inserts time buffers into the schedule that can be used to handle uncertainties.

Orders and operations

All production is customer-order based so that each object in the shop has a customer order with which production orders are associated. The customer order contains information such as release date and due date. The equipment that is maintained in the shop often consists of different parts that are worked on in parallel. Each part has its own production order and the operations of these orders then form a network with precedence relations that describe the work that shall be performed on the object. In most cases the operations on a production order depend on each other and must be performed in a specified order, but exceptions exist where it is possible to perform some operations on the same order independently of each other. Each

operation must be performed in a specified work center and also requires specified materials. See Figure 6.1.

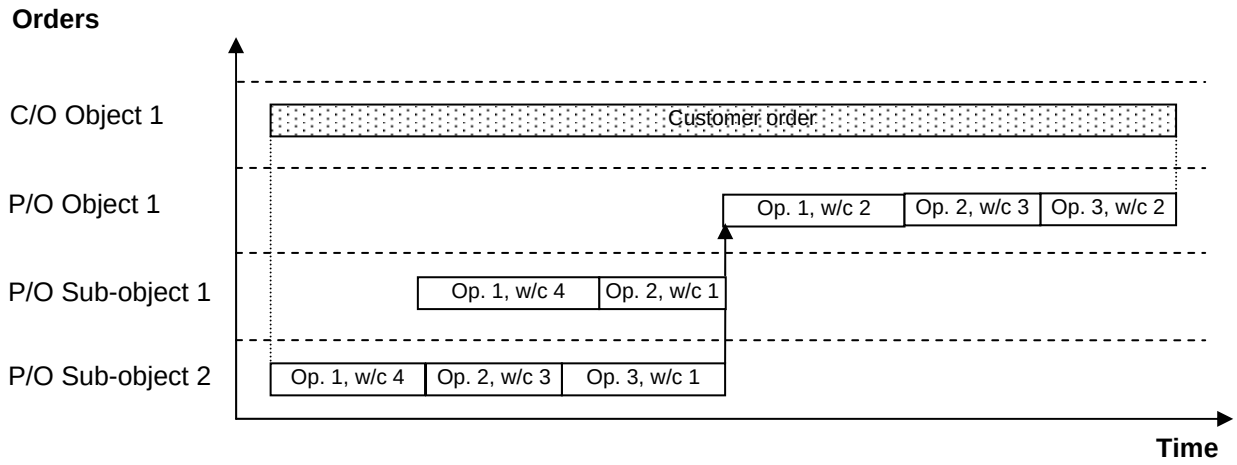


Figure 6.1. The customer order and production orders for an object.

In Figure 6.1, *Object 1* has a customer order and consists of two sub-objects. Operation 1 on the production order for *Object 1* is an assembly operation that uses *Sub-object 1* and *Sub-object 2*.

All information concerning work centers, resources, orders and operations are described in the ERP system and are loaded into the freestanding scheduling system. The scheduling is performed in the freestanding system and then the produced schedule information updates the ERP system.

6.1.3 Organization

The organization consists of six main roles that collaborate in the scheduling process. These are sales department, scheduler, purchasing department, job shop manager, work center managers and production personnel. Sales support and the scheduler belong to the sales support department and job shop manager, work center managers and production personnel belongs to the production department. The tasks and responsibilities of the roles are as follows:

Sales department

The sales department issues the customer orders and initializes work on objects by releasing production orders. This is done in the ERP system. The sales department is the link between planning and scheduling.

Scheduler

The scheduler performs the actual scheduling and has the global overview of the production. The scheduler loads information into the scheduling system, performs the scheduling and updates the ERP system.

Job shop manager

The job shop manager is responsible for all production personnel in the shop and maintains the resource capacity information and also provides feedback on schedule execution to sales support and scheduler as well as performing some manual detailed scheduling. The job shop manager mainly works in the ERP system and views reports in the freestanding scheduling system.

Work center manager

Each work center has a manager that organizes the production and also performs the production together with the production personnel. The work center manager provides feedback on schedule execution to the job shop manager and sometimes direct to the sales department and scheduler. Some detailed manual scheduling is also performed by this role.

Production personnel

The production personnel execute the operations as determined by the schedule. They also provide feedback on schedule execution to the job shop manager and sometimes direct to sales support and scheduler. The production personnel reports performed work in the ERP system.

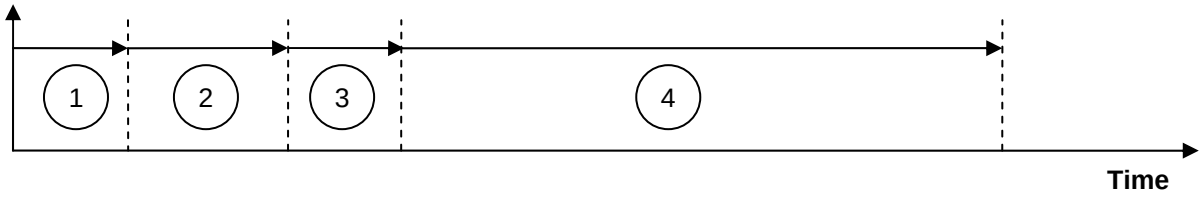
Purchasing department

The purchasing department purchases material required by the production orders and maintains information on available materials. The purchasing department works mainly in the ERP system and views reports in the freestanding scheduling system.

6.2 The scheduling process

6.2.1 The process for an order

The process for a specific object starts when sales support issues a customer order in the ERP system which initializes work on the object. Because some initial work is needed to determine the scope of work on the object a production order with just a few operations to do this is released. When the scope of work is determined the required production orders are prepared to describe the work to be performed. The production orders can now be used to determine the due date by simulating them into the current schedule. This also involves contacts with vendors of spare parts and negotiation with the customer. The simulation is performed by the scheduler. When the due date is set in agreement with the customer the production orders are released and execution of operations is started. As the work progresses operations are reported as finished and information on deviation from the schedule are reported by the production personnel through work center managers and job shop manager to the scheduler and sales support. Depending on the deviations, actions may be needed. This can involve further investigations and renewed contacts with the customer and result in added operations and/or materials. If the due date can not be kept the customer must be informed by the sales department and a new due date must be negotiated. When all operations are finished the production and customer orders are reported as finished and the object is delivered to the customer. See Figure 6.2.



1. Sales department initializes customer order and releases production orders to determine scope of work. The scheduler adds the production orders to the current schedule.
2. The operations on the production orders are executed and the scope of work is determined.
3. The production orders are prepared to reflect the scope of work. A due date is determined by simulating the order in the current schedule. The sales department negotiates with the customer and firms the due date.
4. Operations are executed and deviations from the schedule are reported. Actions to handle deviations are taken in the form of added operations and materials. When all operations are finished the object is delivered to the customer.

Figure 6.2. The order process for an object.

6.2.2 Scheduling

The scheduler makes a new schedule every day. This involves monitoring the progress of operations against the last schedule and reacting to the feedback from the shop floor and to information from the purchasing and sales departments. Events that affect scheduling takes place continuously. Such events are for example operations that take more time than expected, material that is expected to arrive but does not, and objects that have to be rushed to satisfy changing customer needs. The gathered information is then used to create a feasible schedule while also trying to satisfy the scheduling objectives of the shop. See Figure 6.3.

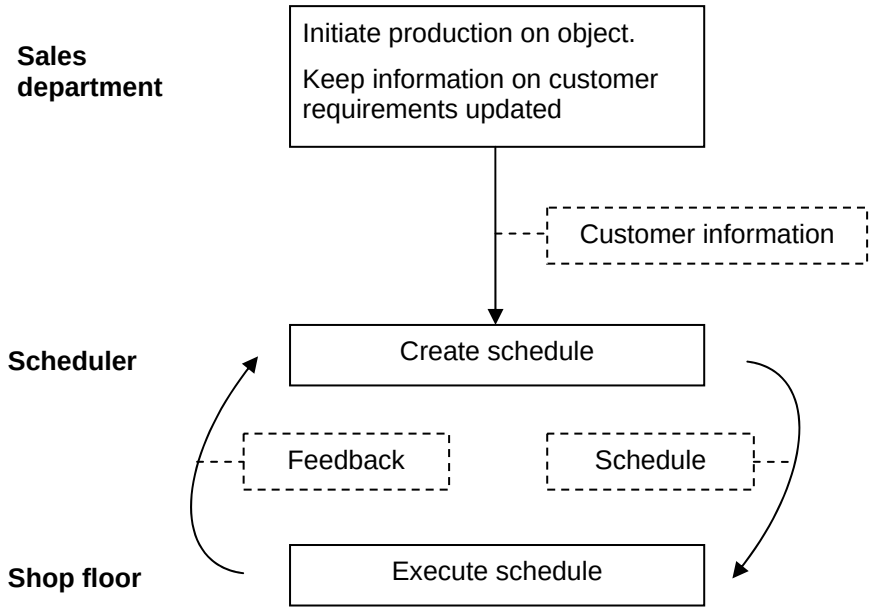


Figure 6.3. The scheduling loop.

Some manual scheduling is also performed on the shop floor by the job shop and work center managers to cope with all the constraining details that cannot be handled by the scheduling system. Since this type of production always involves uncertainties, some scheduling decisions are left to the shop floor. This can for example be that all resources in a work center do not have the same skills and therefore some operations have to be assigned specific resources. The aim is that all resources in a work center shall be interchangeable but this is not always possible due to the diversity of work that has to be performed.

6.3 Scheduling objectives

The most important scheduling objective in this job shop is to keep due dates. Since the objects that are maintained and/or repaired are often critical equipment for the customer, timely deliveries are very important. Secondary objectives are to minimize work in progress and to utilize resources effectively but this should never be at the expense of missing due dates.

6.4 Scheduling tasks

To develop a more general and flexible description, the scheduling process is divided into tasks that are independent of the roles that currently perform them. The task can then wholly or in part be performed by any role. This makes it possible to change the process independently of the system framework. In the following the tasks and what is required to perform them are described. These tasks can also been seen as requirements on a scheduling system for this production environment. Such a system should support these tasks.

6.4.1.1 Generate schedule

The initial schedule should be generated using information from the ERP-system.

6.4.1.2 Manipulate scheduling data during development of schedules

During the development of the schedule it should be possible to manipulate for instance work center capacities and due dates of orders.

6.4.1.3 Prioritize an object, or groups of objects

Priorities should be possible to set for an individual object or groups of objects. Objects should be possible to group by for instance type or customer.

6.4.1.4 Find required, unavailable materials

Materials that are not available when needed for an operation, and thus making it late, must be easily found. This can be accomplished with reports listing those materials.

6.4.1.5 Find over utilized work centers

Work centers that do not have the required capacity to satisfy the needs of operations must be found. It is important to have a way to determine what time periods the work centers are over utilized and how much capacity that is missing.

6.4.1.6 Finding operations that are not executed as scheduled

If an operation is not executed as scheduled it should be possible to easily find this operation.

6.4.1.7 Simulate different scheduling scenarios

Simulating different scheduling scenarios involves for instance changing due dates and priorities of orders and to change available capacity in work centers and available materials.

6.4.1.8 Determine feasible due date for new objects

When a new object is being added to the schedule, it should be possible to put it into the current schedule to determine a feasible due date. When doing this it should also be possible simulate different scheduling scenarios.

6.4.1.9 Compare different schedules

To determine what effects changes to for instance available capacity of work centers have, some way of comparing schedules is required.

6.5 Scheduling algorithm

The scheduling algorithm employed by the scheduling system is a job-based finite capacity method (See 3.5.3.3) that makes no attempt at any optimization of the developed schedule. The basis for the algorithm is a dispatching rule that uses due dates of customer orders together with priorities that are manually set (see section 3.5.2, Earliest Due Date rule). The manual priorities are also set per customer order. The dispatching rule, due date and possible manual priority, impose a priority order on the customer orders and this ordering is strictly followed when developing the schedule. The customer orders are scheduled one by one in the imposed order and the algorithm works by letting the operations belonging to the currently scheduled order allocate capacity in the required work centers and allocate required material. The system maintains two schedules, the finite capacity schedule to be executed on the shop floor and, as a reference, a backwards scheduled infinite capacity schedule. The infinite capacity backwards schedule is the optimal schedule in this environment. In this schedule the latest start and end for each order and operation are maintained and these are used for comparing how far the finite capacity schedule deviates from this optimum.

The system allows for many parameters to be set that affect the way the schedule is developed. The most important are if the algorithm uses forward or backward scheduling. This can be set both as overall parameters for late and non late orders and also for each individual order. It is also possible to, for each work center, set if capacity should be seen as infinite or finite.

In the job shop studied, by experimentation and experience gathered, the parameters have been set so that the developed schedule is forward scheduled and all work centers have finite capacity. This results in a schedule that should be feasible and where every operation is scheduled as early as possible. The decision that operations should be performed as early as possible is a way to cope with the uncertainties in the environment. The reasoning behind this

is that there is greater chance of handling problems occurring during execution of orders without affecting delivery if those problems are detected as early as possible. As described in section 6.1.1, it is common that additional work on an object is discovered during execution.

6.6 Discussion

What makes this environment different from many others is that the work performed is maintenance, not manufacturing of new products. Since this type of production is more unpredictable, the uncertainties caused by this must be handled when developing the schedules. This is accomplished by using a forward schedule where every operation is performed as early as possible, as described in section 6.5, and by setting the utilization rate of work centers to less than 100% as described in section 6.1.2.

When comparing the information gathered in this case study with the background given in chapter 3, it is apparent that production scheduling as performed in this shop, and the problems encountered when doing it, are fairly general. Two important aspects that have been central when implementing the scheduling system in this environment are to make sure the information in the ERP-system is correct and to get sufficient feedback from the shop floor.

Since the information used by the scheduling system comes from the ERP-system, this information must be correct and always maintained to reflect the state of the environment. Because the level of detail and correctness in the information required when developing finite capacity schedules is much greater than what is mostly the case in ordinary ERP-system use, developing this is something that requires much effort when a scheduling system is implemented. This also involves the organization of the company since it is central that all relevant information that is required to develop the schedules is updated in the ERP-system.

Getting sufficient feedback from the shop floor is very important to be able to monitor the progress of schedule execution and also to react to disturbances that occur and to find problem areas. Connected with this is the requirement that the schedules are actually followed in the shop. Getting the shop floor both to follow the determined schedule and to give feedback on it must be considered when implementing a scheduling system. This involves education of the staff and also to make the schedule available on the shop floor and to facilitate feedback by for instance making computers readily available.

The algorithm used in this environment is relatively simple but considering the level of uncertainty that exists it seems appropriate and also in accordance with the rule related in section 3.5. The primary reason for using the system is to get better overview of the

production and to find problem areas, any optimization of the production using more advanced scheduling algorithms can only occur after this is accomplished.

Some functionality that is missing from the current system or could be improved upon has been found during the study. Two of them concern the scheduling tasks in section 6.4. These are the possibility to find operations that have not been executed as scheduled and finding over utilized work centers. Finding operations that have not been executed as scheduled is today manual work; no support exists for this in the scheduling system. Some sort of report that lists those operations should exist. Reports for finding over-utilized work centers exist in the system but it is felt that the granularity of these is not sufficient, since finding the exact work center that causes trouble requires some manual work. Another aspect that could be improved upon is the reports available at the shop floor. These are in the form of dispatching lists, but it would also be useful with more pictorial representations like Gantt-charts to aid in the communication of the schedule.

7 A scheduling system framework

The purpose of this chapter is to propose a scheduling system framework that primarily satisfies the needs of the production environment investigated in the case study, but which can also be more generally used. A prototype including the most important parts of this framework is implemented.

The reason for focusing primarily on satisfying the needs of the production environment in the case study is to keep a manageable scope and to maintain a practical focus and actually be able to implement a prototype that can be tested in reality. The approach will be to first look at the general requirements and then to focus on those requirements that concern the production environment in the case study. The design is object-oriented and the prototype is implemented in Java.

7.1 Requirements

From the information in chapters 2 – 5 and with the architecture described in section 5.2 as a base, requirements for a scheduling system has been gathered. The requirements are grouped into three categories as follows:

Data access and modeling of production environments (see section 7.1.1)

- Possibility to interact with different data sources
- Flexible and adaptable modeling of production environments

Algorithms (see section 7.1.2)

- Possibility to implement and apply different scheduling algorithms

User interaction (see section 7.1.3)

- Possibilities for the user to control the scheduling algorithm and to manipulate the schedule
- Visualization possibilities via reports and other graphical representations
- Evaluation and comparisons of schedules through metrics

In the following three sections these requirements are described in more detail and compared to the specific requirements from the case study in chapter 6.

7.1.1 Data access and modeling of production environments

The interaction with different data sources and modeling of production environments are related. It must be possible to load data from different sources (such as ERP and MES systems) and then to transform the data for use in the model. When the schedule has been developed and the model contains this schedule, the relevant information in the model must be transformed back so that it can update the data sources.

The model must reflect the production environment in enough detail so that it is possible to use the model to develop a schedule that is feasible to execute in reality. The system should have modeling capabilities to do this in a flexible way. How the model is represented is highly dependent on which scheduling algorithms that are going to be used.

7.1.1.1 Requirements from the case study

All production data in the case study comes from the ERP-system and accessing this should not pose any problems. The model must contain the information described in section 6.1.2. This also corresponds to the general description in section 3.1. What must be considered in the model is the information needed for the required reports as described in section 6.4.

7.1.2 Algorithms

The framework should make it possible to implement and apply algorithms that are appropriate for the production environment. The algorithms may come as components or as libraries from other sources and be integrated into the framework.

7.1.2.1 Requirements from the case study

The scheduling algorithm that will be required is one similar to that described in section 6.5. This is basically an *Earliest Due Date* dispatching rule combined with finite capacity scheduling.

7.1.3 User interaction

The requirements possibilities for the user to control the scheduling algorithm and to manipulate the schedule, visualization via reports and other graphical representations and evaluation and comparisons of schedules through metrics are grouped together as user interaction. These requirements are those described in section 5.2.1.3.

7.1.3.1 Requirements from the case study

These requirements are the following (see section 6.4):

- *Generate initial schedule.* This involves loading data from the ERP-system and applying the scheduling algorithm.
- *Manipulate scheduling data during development of schedules.* Scheduling data to be manipulated are work center capacities and due dates of orders.
- *Prioritize an object, or groups of objects.* Functionality to set priorities must exist.
- *Find required, unavailable materials, over utilized work centers and operations that are not executed as scheduled.* Appropriate reports for this must exist.
- *Simulate different scheduling scenarios and compare different schedules.* This is performed by manipulating scheduling data, applying the algorithm and storing the schedule in the scheduling database. Reports where information and metrics from different schedules is shown must exist.
- *Determine feasible due date for new objects.* To do this it must be possible to identify new objects, for instance via timestamps. By using information on which objects are new, manipulation of scheduling data and application of the algorithms, due dates can be determined.

7.2 Design

From the requirements in section 7.1, a structure for a scheduling system framework is shown in Figure 7.1. This basically correspond to the architectures described in [25][37] . The framework has a layered architecture and the objective is to have these layers as loosely coupled and as independent of each other as possible with well defined interfaces and responsibilities. From the bottom up the layers are: Data Access, Extraction/Transformation/Loading/Updating, Scheduling Core and Interfaces/API. The modules grouped together in the Scheduling Core layer, Model, Algorithms, Metrics and Reports, are the ones that perform the actual scheduling and these modules are the focus of the prototype. To make the Scheduling core layer as loosely coupled to the underlying layers as possible it is important that it has no knowledge of how the model is loaded with data or how the data is transferred back to the data sources after scheduling, all this should be provided transparently by the Extraction/Transformation/Loading/Updating layer. At the

other end, the Interfaces/API should insulate the Scheduling core layer by providing a consistent interface for higher layers.

Since the scheduling framework is intended to be used in ComActivity's solution (see section 1.2), the Data Access layer and parts of the Extraction/Transformation/Loading/Updating layer can be provided by that solution. This is also the case with user interfaces for controlling the scheduling application and giving feedback via reports etcetera. The details of this are left outside this dissertation since it is mostly standard procedures that are general and not specific to this application.

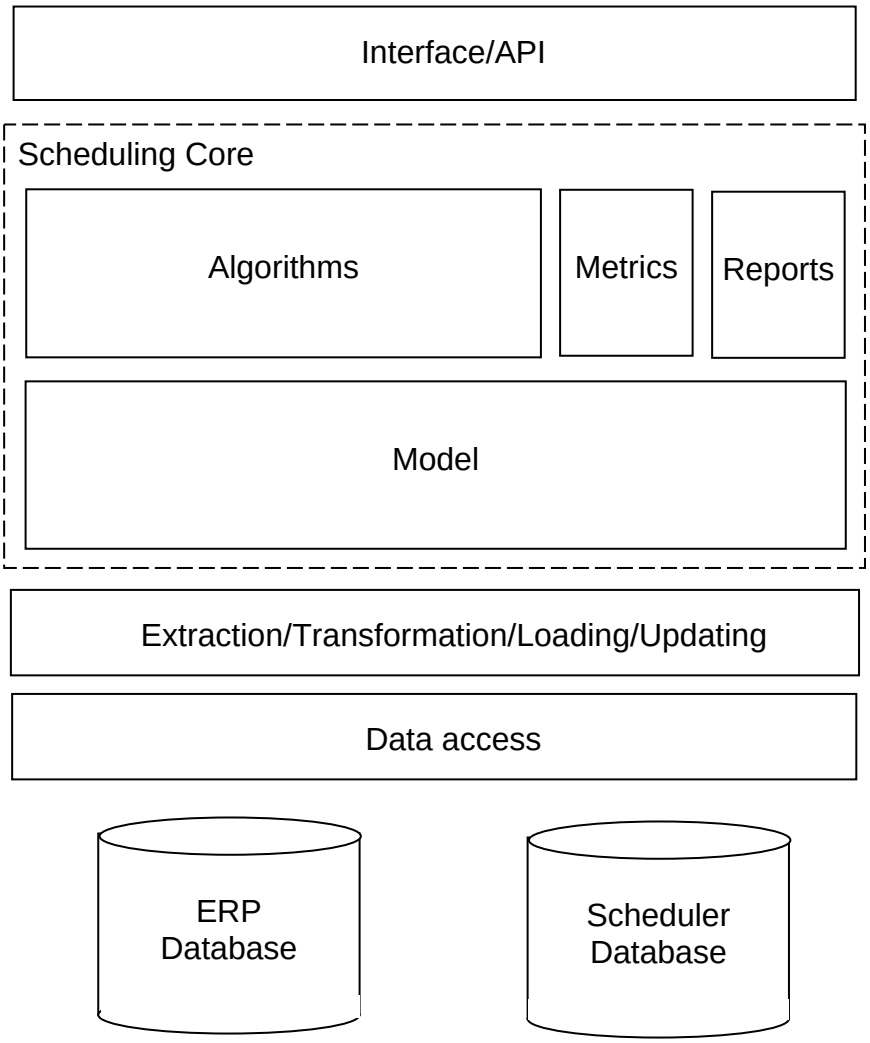


Figure 7.1 Overview of a scheduling system framework

Figure 7.1 an ERP-system database is shown as an example since this is the most common, other data sources can be added as required.

To be able to store data that is used by the system, such as parameters and different versions of the schedule, a database is required. In

Figure 7.1 this is called *Scheduler Database*.

In the following sections each module of the framework is described.

7.2.1 Data access

Data access should pose no problem. Connecting to, and accessing data from, databases are standard procedures.

7.2.2 Extraction/Transformation/Loading /Updating

The production data must be extracted from the data sources, transformed into a suitable format and then loaded into the model. When updating the data sources with information from the developed schedule, the data in the model must be transformed back to an appropriate format that can be used to update the data sources.

The extraction involves executing queries against the appropriate tables in the source databases to retrieve the required information. The data models in these data sources and the model used in the scheduling system are not the same; therefore a transformation must be performed. This transformation also makes the model and the data sources independent of each other. The design of the model and the design of the data sources can vary independently as long as the transformation is changed accordingly to provide the required mappings between these.

Updating the data sources with transformed scheduled data from the model, like the extraction, involves executing queries against the appropriate tables.

The details of this layer are not further related in this dissertation since those details are highly dependent on the data sources used.

7.2.3 Model

The model is arguably the most important module of the framework. The first criterion it must fulfill is to be a sufficiently detailed representation of the production environment to make a schedule that is feasible in the model also feasible in reality. It must also be possible to apply different algorithms to the model. In addition to this the model must be able to provide

information that makes it possible to develop metrics and reports that satisfy the different requirements described in section 7.1.3.

The approach taken in this framework is to provide a base model that is as simple as possible and consists of the parts described in section 3.1, production resources, orders and operations and materials and subparts. This basic model is primarily a data structure that stores information and contains as little logic as possible. By keeping the model passive and putting all logic in the upper layers, such as algorithms, metrics and reports, the framework can provide flexibility and support different scheduling requirements.

Since different types of algorithms require different model representations as described in section 4.2 there must be some way to extend the model to provide for this. For instance a genetic algorithm requires some lightweight representation of the model to store the population of schedules while a constraint programming approach needs a model that can be utilized by a constraint programming toolkit. Figure 7.2 shows a way to allow for different model representations. The basic model is extended with modules in which the model's information is transformed into the representations that are required for the particular algorithms used. An application of this can be for example the combination described in section 4.3 where dispatching rules are used to develop an initial schedule that is then refined by simulated annealing. Except for the algorithms no part of the framework should have to access the representations directly, only via the basic model. This makes the algorithms, and their required representation, transparent to the rest of the framework. In Figure 7.2, the metrics and reports modules use the basic model to get their required information.

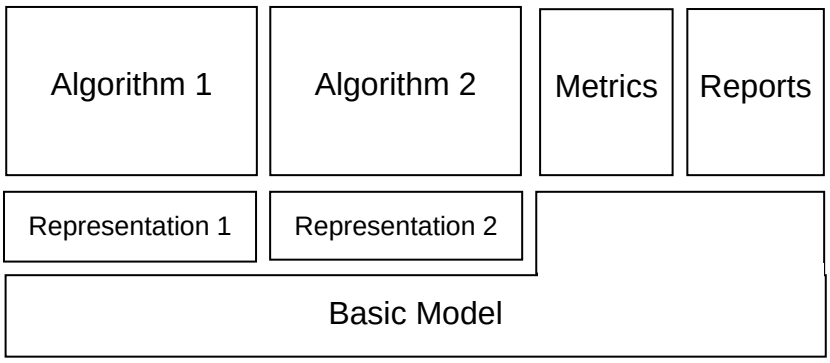


Figure 7.2. The Basic Model with extensions for different representations

It is also possible to use the scheduling components described in section 5.2.2 this way. The scheduling components only need to interact with the basic model and thus are transparent to the rest of the framework. The representation can either be part of the scheduling component and just fed from the basic model or fully provided by an extension of the basic model. For the algorithm required for the prototype, the basic model is sufficient, no extension is needed.

In the following sections the different parts of the model are described, UML class diagrams [11] are used to describe class structures when this is required. These diagrams are kept as simple as possible and are not complete descriptions of the classes.

7.2.3.1 Handling of dates and times

Much of the information that is handled when scheduling involves dates and times, it is therefore important to provide for this in the model. The information that must be possible to handle are date and time of day (e.g. start date and time of an order), time duration (e.g. duration of an operation) and date/time intervals (e.g. available time in a production resource).

Time duration

Durations in this system are measured in hours and minutes. In manufacturing systems, such as the one described in the case study, it is common to use time periods that are 1/100 of an hour instead of minutes. Since time durations are used to manipulate date/time values it is much more convenient to use an hour minute representation. Therefore the Extraction/Transformation/Loading/Updating layer (see section 7.2.2) must perform necessary conversions so that the model is loaded whit durations represented as hours and minutes.

A class called *TimeUnit* is used to represent duration. *TimeUnit* provides methods for addition, subtraction and multiplication of time durations as well as for adjusting durations by the utilization rate of resources.

Date/time

Date/time is handled by a class called *DateTime* which represents a point in time by year, month, day, hour and minute. *DateTime* provides methods to add and subtract *TimeUnit*'s from the represented date/time as well as other necessary date/time manipulations.

Date/Time interval

The representation of a time interval is useful when manipulating schedules. This is provided by a *TimeInterval* class. This class contains start and end *DateTime*s and methods to determine the relation to other *TimeIntervals* (such as before, inside or after). A useful discussion of time intervals that is often referenced in papers concerning scheduling is [2].

7.2.3.2 Production resources

Production resources in the case study consist of work centers and work center resources. The available capacity in those resources is described by a calendar, shifts and capacity adjustments. The calendar describes which dates production is performed, for example that no production is performed on the Christmas holiday. Shift describes which days and hours a resource is open, for example Monday to Friday from 7:00 to 16:00. Capacity adjustments describe temporary adjustments to the capacity described by the calendar and shifts, for example that a particular resource is working overtime a certain date.

Work centers and work center resources

The class structure of work centers and work center resource is shown in Figure 7.3. The class *WorkCenters* is a collection containing all work centers and it is through this class the work centers are accessed by other classes. The *WorkCenter* class represents a particular work center and this in turn contains a collection of *WorkCenterResource*'s. Both *WorkCenter*'s and *WorkCenterResource*'s are identified by their names.

The class *UtilizationRecord* is used to store information on how a work center resource is utilized and also to describe when a work center cannot provide the required capacity. The latter is referred to as overload intervals. An *UtilizationRecord* can store information about which time interval it concerns, which operation that is performed during the interval and a collection of *TimePerDayRecords* that describes how much time that is utilized each day. The collections of *UtilizationRecord*'s are intended to be used by algorithms to store information, both for internal use in the algorithms and for later use by the report module.

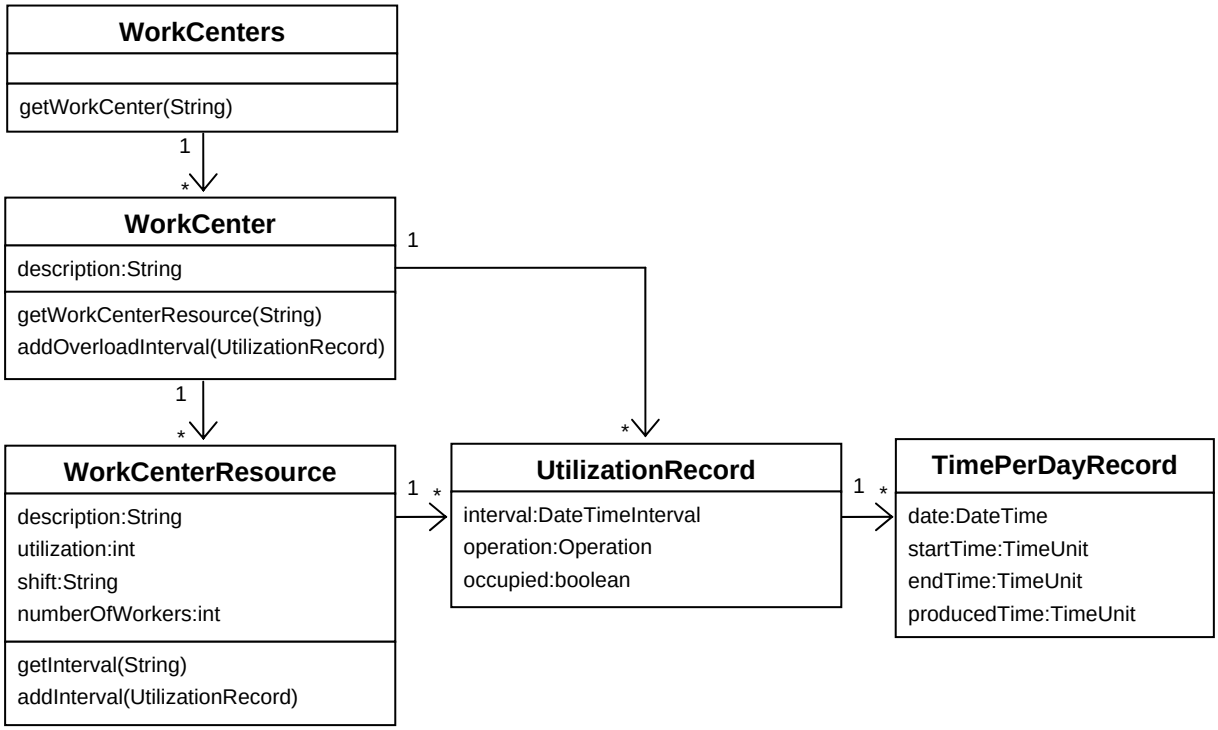


Figure 7.3. Class diagram of production resources

Shifts

The class structure of shifts is described in Figure 7.4. *Shifts* contain a collection of shifts. A *Shift* is identified by its name. Each *Shift* contains *ShiftDay*'s which in turn contains *ShiftDayPart*'s. A typical shift is seven shift day's describing Monday to Sunday where Monday to Friday are production days and Saturday and Sunday are non-production days. Each production day contains shift day parts which describe open hours of the day, for instance one part from 7:00 to 12:00 and another from 13:00 to 16:00 which implies a one hour lunch break between 12:00 and 13:00.

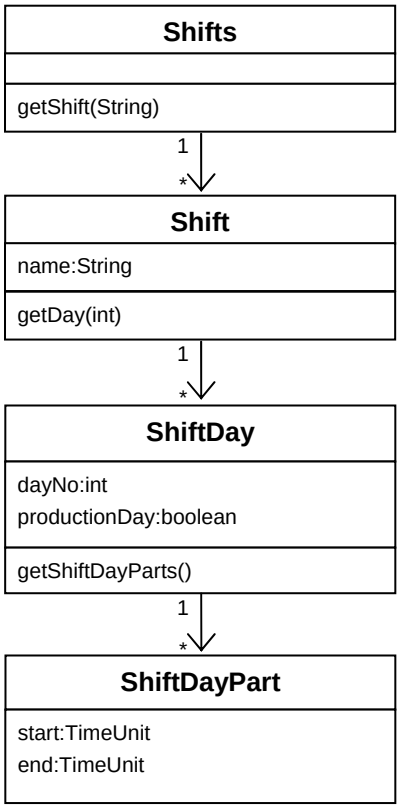


Figure 7.4. Class diagram of shifts.

Calendar

Since most dates are production days the calendar only has to be represented by a list containing the dates of non-production days.

Capacity adjustments

Figure 7.5 shows the structure of the classes describing capacity adjustments. If a capacity adjustment exists for a certain date the information in the adjustment should be used instead of the information described by the calendar and shifts. A capacity adjustment is identified by the combination of which work center resource and what date/time interval it concerns. The capacity adjustments for a particular work center resource cannot have overlapping date/time intervals. A capacity adjustment for a work center resource concerning a specific date must be unique.

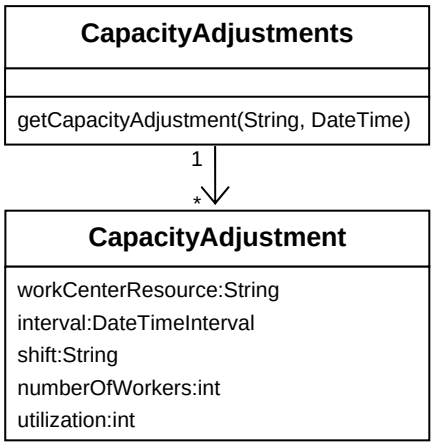


Figure 7.5. Class diagram of capacity adjustments

7.2.3.3 Orders and operations

Operations

The basic entities that describe the work to be performed are the operations. An operation contains information such as in what work center it should be performed, the expected time required to perform it (its duration) and the operations adjacent to it (see Figure 4.1).

Depending on the production environment and the used scheduling method, the requirements on the information contained in an operation can differ. From the case study, required additional information is: required materials, pre- and post operation times, operation status (e.g. started or not started), production time already performed as well as storage of scheduled start and end and scheduled work center resource.

Figure 7.6 shows a class diagram of the operation structure. An *Operation* is identified by *internalOpId* which is a synthetic id produced by the Extraction/Transformation/Loading/Updating layer. The reason for this synthetic id is to provide for easier indexing and sorting of operations. An operation's material requirements are represented by the class *OperationMaterialRequirement*. During the run of the scheduling algorithm, encountered shortages of material can be stored in a list of *MaterialShortageRecord*'s. Different dates for an *Operation* that are encountered during scheduling, such as unconstrained and constrained start and end, can be stored as key-value pairs in a list called scheduling dates.

The class *Operations* provides access to a collection of operations.

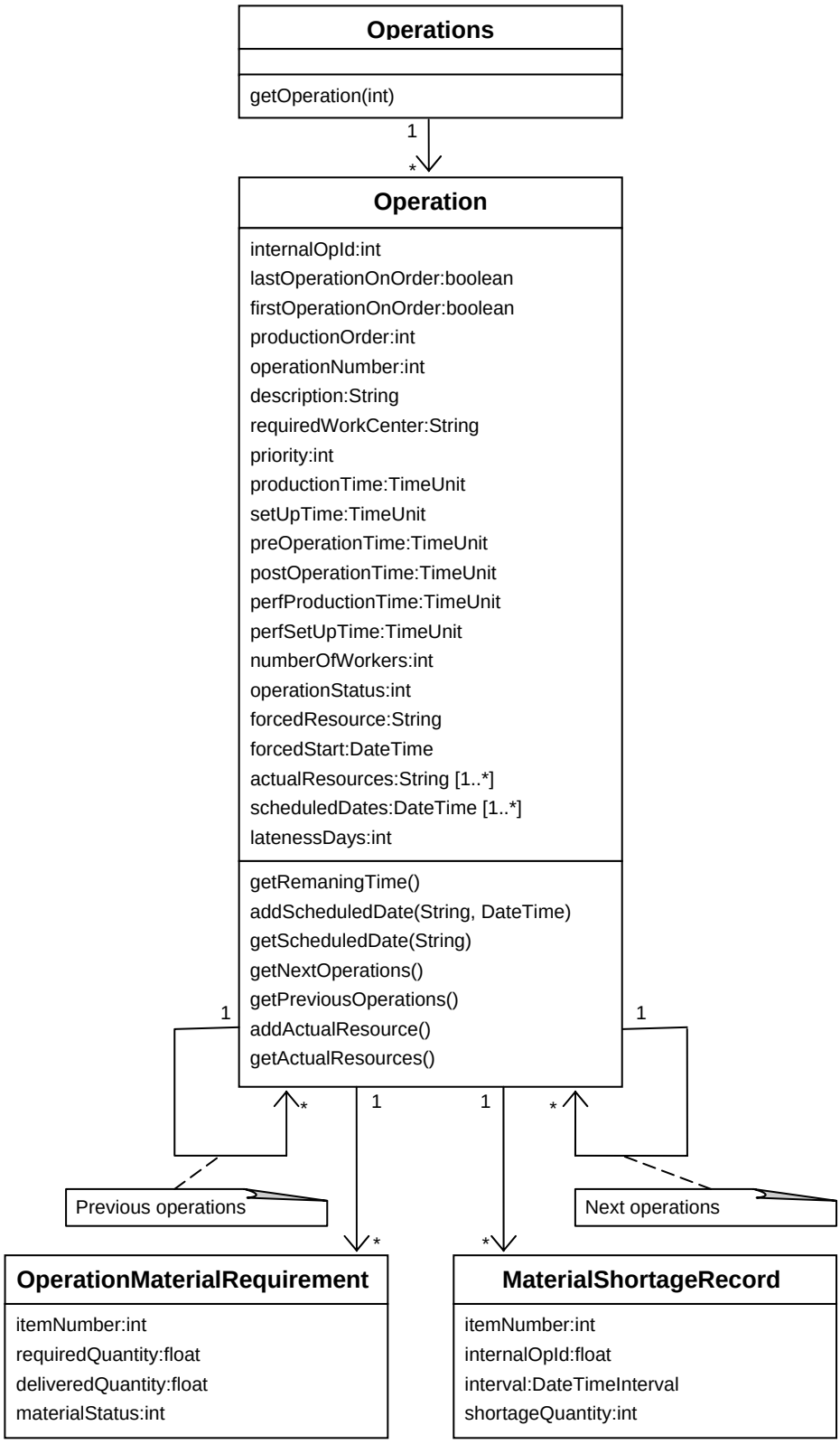


Figure 7.6. Operations class diagram

Orders

Orders can be seen, at least in the context of scheduling algorithms, as groupings of operations. The hierarchy of customer order, customer order line, production orders and operations are shown in Figure 7.7. This can be compared with Figure 6.1. The customer order in Figure 6.1 actually consists of a customer order and a customer order line.

For scheduling as it is performed in the environment described in the case study, the most important entities are customer order line, production orders and operations. The customer order line is associated with one production order representing an object that is to be repaired in the shop. This order can then be associated with sub production orders representing sub objects (see Figure 6.1). The production order associated with the customer order line is called the top production order. One of the operations contained in this order is an assembly operation that requires the sub objects represented by the sub production orders. Sub production orders contain information on the top order and the assembly operation in this order that they are associated with in the *nextLevelOrder* and *nextLevelOperation* attributes. As in the Operation class the order classes also can store key-value pairs of scheduling dates.

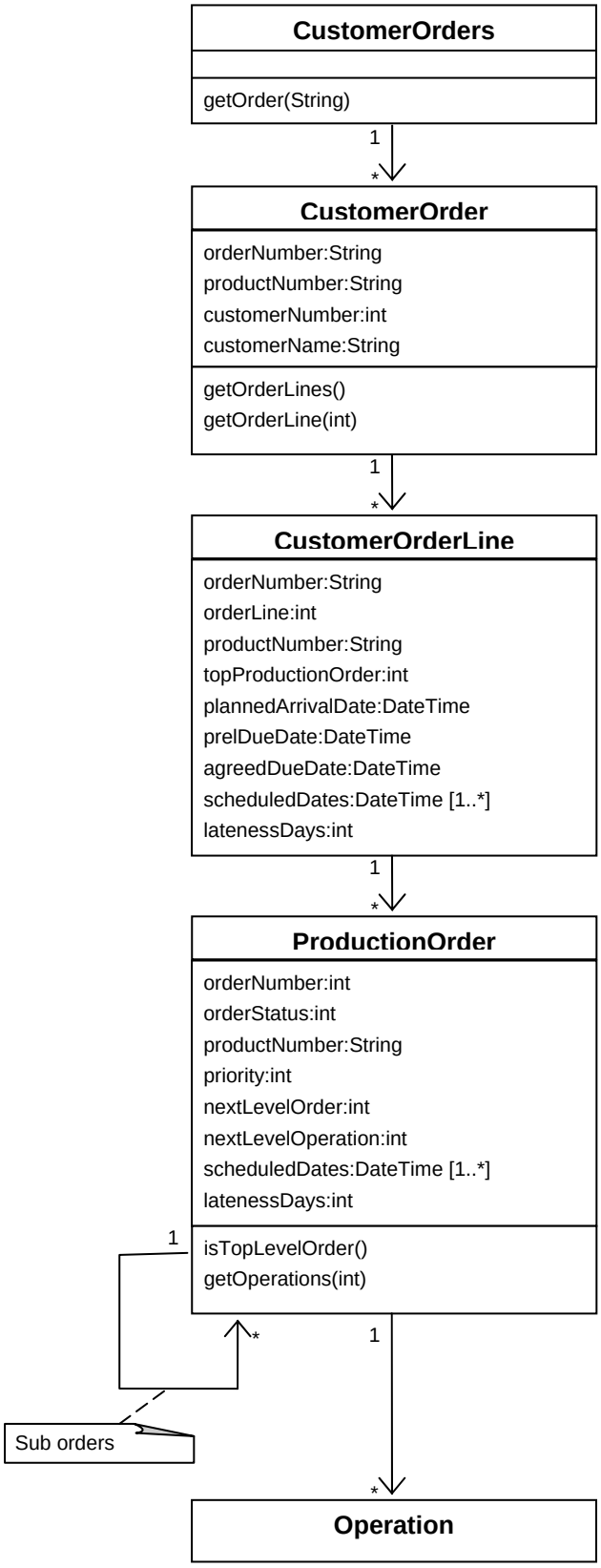


Figure 7.7. Orders class diagram

7.2.3.4 Materials and subparts

Materials and sub-parts are handled the same way in this model, both are described as materials. The class structure for materials is shown in Figure 7.8. A specific material is identified by its item number. Initially a material has an on hands balance, which is the quantity available in stock, and possibly pending deliveries from the supplier. The attribute *leadTimeDays* is the number of days required to get the material in stock either when ordered from a supplier or manufactured in the company. The class *MaterialRequirement* is intended to be used by algorithms to keep track of which operations that require the material and by which date it is required.

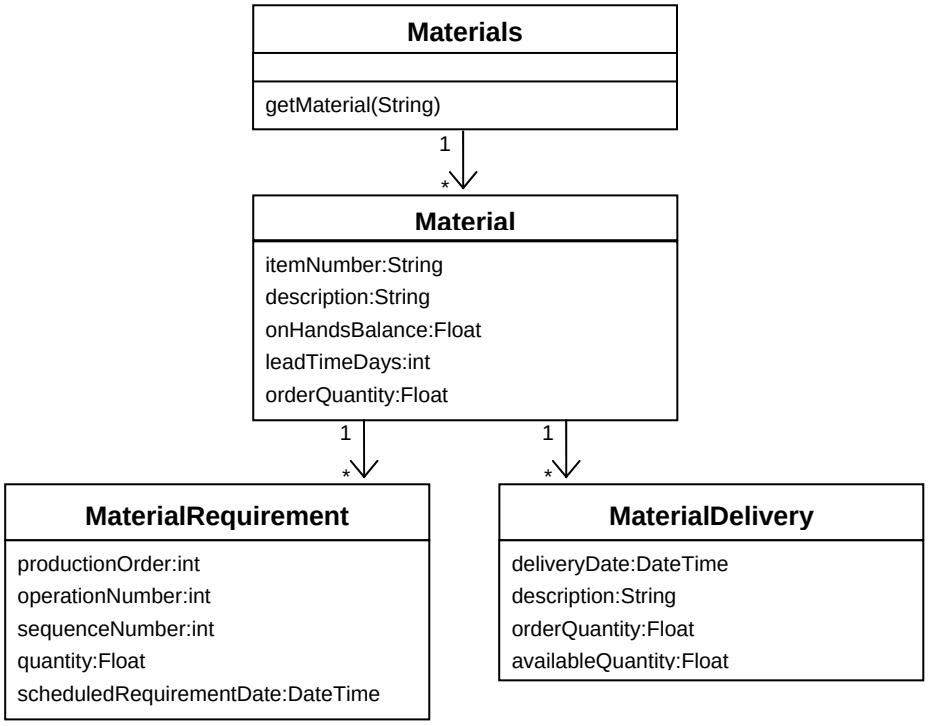


Figure 7.8. Class diagram of materials

7.2.4 Algorithms

The algorithms required for the prototype should utilize a dispatching rule using the due dates of customer order lines to determine priority, in combination with considering the constraints imposed by work center capacities and available materials. The developed schedule should be forward scheduled, that is every operation is to be executed as early as possible.

In this section pseudo-code with some java-like constructs is used to describe the algorithms.

7.2.4.1 Algorithm primitives

To provide for modularity and easier implementation of algorithms, a set of algorithm primitives is defined. This set contains methods for getting the earliest end of a list of operations given a start date/time, for getting the latest start of a list of operations given an end date/time and for getting the earliest available date for a list of materials. It is in these algorithm primitives the actual finite capacity scheduling is performed since they operate on the lowest level and therefore in detail must consider all constraints imposed on the production environment.

Getting earliest end of an operation given a start date/time

When forward scheduling, a start date/time is known and an end date/time is to be determined. An operation has a duration time, pre- and post operation times and a required work center it should be performed in. The work center consists of a number of similar work center resources that can perform the operation. Each work center resource has a utilization rate, a shift associated with it, possibly capacity adjustments and a list of utilization records describing available and occupied intervals of the resource. Given this information the problem is to determine in which of the work center's resources the operation should be performed to get the earliest end date/time. An algorithm to do this is shown in Figure 7.9. For each work center resource the intervals describing the time available to perform work are used to determine when the operation can be performed. This is done by checking the calendar, the shift associated with the resource and possible capacity adjustments of the resource. The start and end of the operation that is derived from the work center's resources are compared and the operations actual resource is set to the resource giving the earliest end.

ALGORITHM TO DETERMINE END OF AN OPERATION WITH GIVEN START

```
function getEarliestOperationEnd(start, operation)
begin
  resources = operation.workCenter.getWorkCenterResources
  for each resource in resources do
    requiredOpTime = operation.duration/resource.utilization
    availableIntervals = the resources utilization intervals
                         that are not occupied
    order availableIntervals using start, from earliest to latest
    while availableIntervals.hasMoreElements do
      availableInterval = availableIntervals.getNextInList
      if availableInterval.end after start and requiredOpTime can
         be performed in availableInterval then
        if start after availableInterval.start then
          tempStart = start
        else
          tempStart = availableInterval.start
        end if
        tempEnd = operation end calculated using requiredOpTime
      end if
    end
    if currentEarliestEnd is set then
      if tempEnd before currentEarliestEnd then
       currentStart = tempStart
       currentEarliestEnd = tempEnd
       currentEarliestResource = resource
      end if
    else
      currentStart = tempStart
      currentEarliestEnd = tempEnd
      currentEarliestResource = resource
    end if
  end
  operation.start = currentStart
  operation.end = currentEarliestEnd
```

```
    operation.actualResource = currentEarliestResource
    return
  end
```

Figure 7.9. Algorithm to determine earliest end date/time of an operation when the start date/time is given.

Getting latest start of an operation given a end date/time

Backwards scheduling of an operation is performed by from a given end date/time and determining the latest start date/time of the operation. An algorithm for this is shown in Figure 7.10. The basic structure of the algorithm is like the one in Figure 7.9 but the available intervals are searched from the latest end to the earliest using the given end as a starting point.

```
ALGORITHM TO DETERMINE START OF AN OPERATION WITH GIVEN END

function getLatestOperationStart(end, operation)
begin
  resources = operation.workCenter.getWorkCenterResources
  for each resource in resources do
    requiredOpTime = operation.duration/resource.utilization
    availableIntervals = the resources utilization intervals
                         that are not occupied
    order availableIntervals using end, from latest to earliest
    while availableIntervals.hasMoreElements do
      availableInterval = availableIntervals.getNextInList
      if availableInterval.end before or equal to end and
         requiredOpTime can be performed in availableInterval
         then
        tempEnd = end
        tempStart = operation start calculated using
                    requiredOpTime
      end if
    end
    if currentLatestStart is set then
      if tempStart after currentLatestStart then
       currentLatestStart = tempStart
       currentEnd = tempEnd
       currentLatestResource = resource
```

```
      end if
    else
      currentLatestStart = tempStart
      currentEnd = tempEnd
      currentLatestResource = resource
    end if
  end
  operation.start = currentLatestStart
  operation.end = currentEnd
  operation.actualResource = currentLatestResource
  return
end
```

Figure 7.10. Algorithm to determine latest start date/time of an operation when the end date/time is given

Determine earliest available date of materials required to perform an operation

The materials required to perform an operation have to be available when the execution of the operation is started. To do this, the availability of each required material has to be checked. The earliest available date of the materials required to perform an operation is the latest available date found among the required materials. Figure 7.11 shows an algorithm to do this. If a materials on-hands balance is more than the required quantity the available date is today, else the deliveries must be checked to determine if these can satisfy the request (there might be previous allocations from other operations). If a delivery can satisfy the request the available date is this delivery date. If neither a materials on-hands balance nor a future delivery can satisfy the request, the available date is set to today plus the number of days that is the purchasing leadtime of the material. The purchasing leadtime of a material is the time expected to elaps from purchase until the material is available for use on the shop floor.

ALGORITHM TO DETERMINE AVAILABLE DATE OF MATERIALS REQUIRED TO PERFORM A GIVEN OPERATION

```
function getEarliestMaterialAvailableDate(operation)
begin
 opReqMaterials = operation.getRequiredMaterials
 for each opReqMaterial in opReqMaterials do
   material = Materials.getMaterial(opReqMaterial.itemNumber)
   if material.onHandsBalance >= opReqMaterial.quantity then
    material.onHandsBalance = material.onHandsBalance  -
                              opReqMaterial.quantity
    availableDate = today
   else if material delivery exists then
    determine delivery date of delivery that can satisfy the
    requirement.
    Allocate the required quantity in found delivery
    availableDate = delivery date of found delivery
   else
    availableDate = today + material.leadtime
   end if
   if earliestAvailableDate is set then
    if availableDate after earliestAvailableDate then
     earliestAvailableDate = availableDate
    end if
   else
    earliestAvailableDate = availableDate
   end if
 end
 return earliestAvailDate
end
```

Figure 7.11. Algorithm to determine earliest available date of materials required to perform a given operation.

7.2.4.2 Basic algorithm

This section describes a basic algorithm that uses the scheduling primitives described in section 7.2.4.1 to develop a schedule satisfying the requirements in section 7.1.3.1. The steps for developing a schedule using this algorithm are:

1. Backwards schedule all customer order lines from their respective due dates without considering the constraints imposed by work center capacities and availability of materials to get an optimal schedule (the schedule is optimal by the definition given in section 3.4). This schedule is used as a reference to compare other schedules.

   What can be derived from this schedule are those operations and orders that are, or probably are going to be, late even if work centers available capacities and materials availability were unconstrained because derived start and/or end of operations are before a given schedule start date.

2. Forward schedule all customer order lines from a given start date considering the constraints imposed by work center capacities and availability of materials. To find out which orders and operations that are late because of unavailable materials, the start and end of operations are first derived without the constraint imposed by material availability. The earliest material availability date is then checked and compared to the derived start date of the operation. If the earliest material availability date is after the derived operation start date this fact is stored in the operation and new operation start and end date/time's are derived using the earliest material availability date as start parameter.

1. Backwards scheduling using unconstrained capacity

Since this scheduling method decomposes the problem at the order (or job) level as described in section 3.5.3.3, the fundamental operation is to schedule a sequence of operations. Figure 7.12 shows a sequence of operations. An operation in this figure has a duration, a pre operation time and a post operation time. The duration that is used during scheduling is the calculated duration derived from the planned duration of the operation and the resource's utilization rate. If for example the operations planned duration is one hour and the resource's utilization rate is 80%, the duration used when scheduling is 1.2 hours. Pre- and post operation times can be 0 or higher. In Figure 7.12, the operations are executed in order from O_1 to O_n.

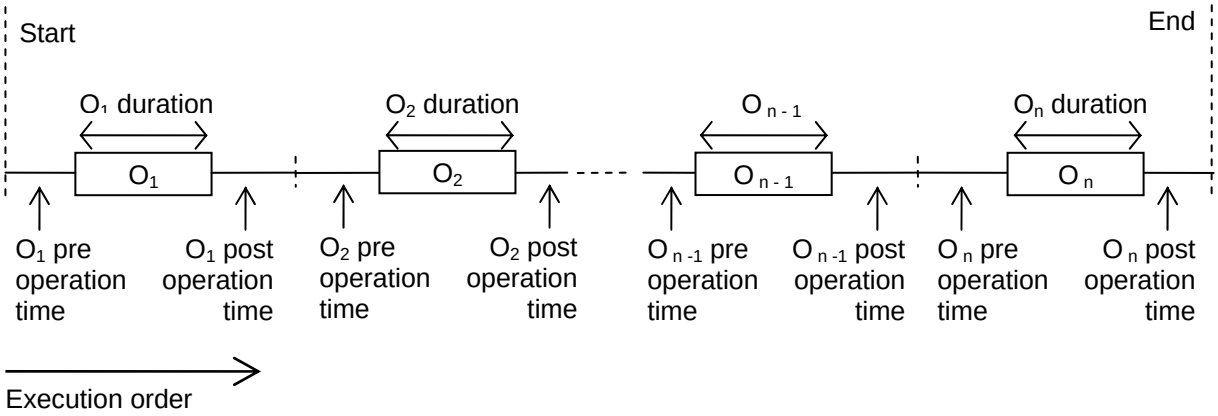


Figure 7.12 Operation sequence

An algorithm to backwards schedule a sequence of operations is shown in Figure 7.13. The algorithm starts with the last operation in the sequence and schedules the operations one by one backwards through the sequence.

```
ALGORITHM TO BACKWARD SCHEDULE A SEQUENCE OF OPERATIONS

function backwardScheduleOperations(endDate, operations)
begin
  sort operations from last to first
  Operation curOp, nextOp
  while operations.hasMoreElements do begin
    curOp = operations.getNextInList
    Date opStart
    if curOp is first in list then
      opEnd = endDate - curOp.postOperationTime
    else
      opEnd = nextOp.start - nextOp.preOperationTime -
      curOp.postOperationTime
    end if
    getLatestOperationStart(opEnd, curOp)
    nextOp = curOp
  end
end
```

Figure 7.13. Algorithm to backward schedule a sequence of operations

Figure 7.14 shows a sequence of operations ordered from last operation to first as used in the algorithm described in Figure 7.13. In the figure, the parameters *endDate* and *operations* are shown together with the variables *nextOp* and *curOp*. The operations are visited from O_n to O_1. The figure shows the state of the variables when the algorithm is in the second iteration of the while loop.

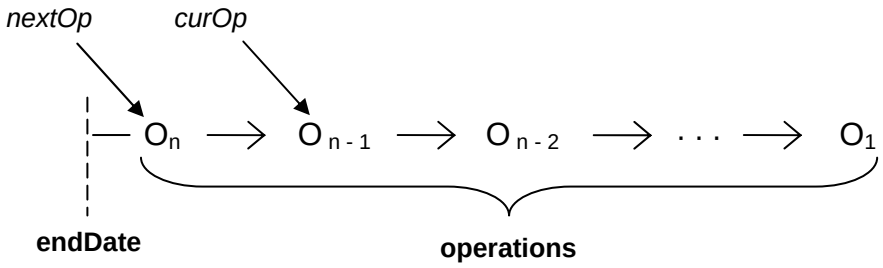


Figure 7.14 Operation sequence as used in Figure 7.13

Figure 7.15 shows how the algorithm described in Figure 7.13 is used to backward schedule a list of customer order lines. The dispatching rule uses the due date of the order lines to set the priority order of jobs to be scheduled.

```
ALGORITHM TO BACKWARD SCHEDULE CUSTOMER ORDER LINES

customerOrderLines = list of customer-order lines to be scheduled
Date schStart (start date of schedule)

Sort customerOrderLines in decreasing due date order

while customerOrderLines.hasMoreElements do begin
  customer-order-line orderLine = customerOrderLines.getNextInList
  production-order topProdOrder = orderLine.getTopProdOrder
  backwardScheduleOperations(orderLine.dueDate,
                             topProdOrder.getOperations)
  topOrder.start = start of first operation on order
  if topProdOrder has sub-order then
    subOrderList = topProdOrder.getSubOrders
    while subOrderList.hasMoreElements do begin
      production-order subOrder = subOrderList.getNextInList
      topOrderOp = topOrder.getOperation(suborder.nextLevelOp)
      subOrderEnd = topOrderOp.start
```

```
      backwardScheduleOperations(subOrderEnd,
                                 subOrder.getOperations)
    end
  end if
end
```

Figure 7.15. Algorithm to backward schedule customer order lines.

2. Forward scheduling using constrained capacity

As described above the fundamental operation when a scheduling problem is decomposed at the job level is to schedule a sequence of operations (see Figure 7.12). When forward scheduling, the problem is to find the earliest end of the last operation in a sequence of operations. To do this the sequence has to be ordered from the first operation to the last and then the operations are visited in order. The first operation's start is given and the other operations get their start from the determined end of the previous operation. In the order structure described in section 7.2.3.3, a top level order has one operation that is connected to the sub-orders. The end date of the latest sub-order must be considered when scheduling this operation. An algorithm for this is shown in Figure 7.16. This algorithm satisfies the constraint imposed by work center capacities by storing the intervals needed to perform the operations as occupied intervals in the resources. The constraint imposed by material availability is satisfied by checking the earliest material availability date and reschedule operations if material is unavailable.

ALGORITHM TO FORWARD SCHEDULE A SEQUENCE OF OPERATIONS

```
function forwardScheduleOperations(startDate, operations,
                                   subOrdersConnectOp, subOrdersEnd)
begin
  sort operations in order from first to last
  Operation curOp, prevOp
  while operations.hasMoreElements do begin
    curOp = operations.getNextInList
    Date opStart
    if curOp = subOrdersConnectOp then
```

```
        if curOp is first in list then
          opStart = subOrdersEnd + curOp.preOperationTime
        else
          opStart = max(prevOp.end + prevOp.postOperationTime,
                        subOrdersEnd)
          opStart = opStart + curOp.preOperationTime
        end if
      else if curOp is first in list then
        opStart = opStart + curOp.preOperationTime
      else
        opStart = prevOp.end + prevOp.postOperationTime +
        curOp.preOperationTime
      end if
      getEarliestOperationEnd(opStart, curOp)
      Date matAvailDate = getEarliestMaterialAvailableDate(curOp)
      if matAvailDate after opStart then
        store this fact in the operation
        getEarliestOperationEnd(matAvailDate, curOp)
      end if
      WorkCenterResource resource = curOp.actualWorkCenterResource
      resource.addOccupiedInterval(curOp.start, curOp.end)
      prevOp = curOp
    end
  end
```

Figure 7.16. Algorithm to forward schedule a list of operations

Figure 7.17 corresponds to Figure 7.14 and shows a sequence of operations ordered from first operation to last as used in the algorithm described in Figure 7.16. In the figure, the parameters *startDate* and *operations* are shown together with the variables *prevOp* and *curOp*. The operations are visited from O_1 to O_n. The figure shows the state of the variables when the algorithm is in the second iteration of the while loop.

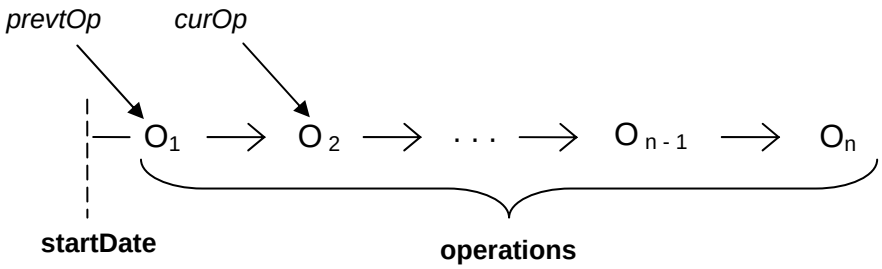


Figure 7.17 Operation sequence as used in Figure 7.16

Figure 7.18 shows how the algorithm described in Figure 7.16 is used to forward schedule a list of customer order lines.

```
ALGORITHM TO FORWARD SCHEDULE CUSTOMER ORDER LINES

customerOrderLines = list of customer-order lines to be scheduled
Date schStart (start date of schedule)

Sort customerOrderLines in decreasing due date order

while customerOrderLines is not empty do begin
  customer-order-line orderLine = customerOrderLines.getNextInList
  production-order topProdOrder = orderLine.getTopProdOrder
  if orderLine.releaseDate after schStart then
    ordStart = orderLine.releaseDate
  else
    ordStart = schStart
  end if
  toStart = ordStart
  if topProdOrder has sub-order then
    subOrderList = topProdOrder.getSubOrders
    while subOrderList.hasMoreElements do begin
      production-order suborder = subOrderList.getNextInList
      forwardScheduleOperations(ordStart, subOrder.getOperations,
                                0, 0)
      set subOrder end date to end of last operation on order +
      postOperationTime of last operation
    end
```

```
    subOrdEnd = latest end date of orders in subOrderList
  end if
  forwardScheduleOperations(ordStart, topProdOrder.getOperations,
                            topOrder.subConnectOp, subOrdEnd)
  set topProdOrder end date to end of last operation on order +
  postOperationTime of last operation
  set orderLine end date to end date of topProdOrder
end
```

Figure 7.18. Algorithm to forward schedule customer order lines

7.2.4.3 Further developments of algorithms

Some details have been left out of the algorithms described in the previous sections. The reason for this has been to focus on the basic structure of the algorithms and not having to deal with special cases and exceptions that occur in a real production environment. Details that must be added when using the algorithms in a real production environment are:

- Handling of started operations. All started operations must be scheduled before operations that are not started. Since an operation can be started anywhere in the operation list of an order if such a opportunity occurs on the shop floor, there can be situations where an operation earlier in the list can get a derived end later than the end of following started operation. To handle this it has to be made sure that an operation cannot start before any of the operations preceding it. Changes to handle started operations are required in the algorithm shown in Figure 7.16.
- Handling of finished production orders. Since some or all of the suborders associated with a top production order may be already finished, this must be handled in the algorithms shown in Figure 7.15 and Figure 7.18.
- Handling of external priorities. This can be seen as an extension of the used dispatching rule. An example can be to first sort the orders by external priority, then by due date.
- Storing information required for metrics and reports that is found during algorithm execution. This can for example be how long an operation must wait in queue before it is executed at a work center or how long an operation is delayed because of missing

materials. Exactly what information that should be stored depends on what is required by the metrics and reports.

- Handling of missing or inconsistent data. In real production environments, situations can occur where the data used for scheduling is missing or inconsistent. This can be handled either in the algorithms, in the Extraction/Transformation/Loading/Updating layer or by a combination of both. How it should be handled depends on the situation but the user must in some way be notified of these problems. The goal should be to detect the problems as early as possible, preferably during extraction and loading. The user can then be notified about the problems and solve these before the algorithms are run. Some inconsistencies can be difficult to find during extraction and loading, therefore the algorithms probably must contain some sort of handling of these exceptions.

For the prototype, the algorithms must handle started operations and finished production orders. Adding this functionality to the algorithms in question should pose no difficulties since no changes to the structures of the algorithms are required. The changes mostly involve performing additional checks at appropriate places.

Handling of external priorities is not implemented in the prototype. The only part of the algorithms effected by external priorities are the sorting of customer order lines. To add this functionality later should be relatively easy.

The metrics and reports information that should be stored by the prototype are when an operation cannot be performed as required because of overloaded work centers and when an operation is delayed because of material shortages.

The data used to test the prototype should not contain any missing or inconsistent information; therefore no handling of this is required.

7.2.5 Metrics and Reports

The responsibility of the Metrics and Reports modules is to provide information that is required to produce reports and other visualizations for feedback to the user of the scheduling system. This information can either be communicated to clients of the framework via the Interface/API layer or stored in appropriate data sources and then retrieved directly from these sources by the client.

7.2.5.1 Metrics

Metrics can be for example the lateness of orders and operations or the utilization of work centers. Metrics can be used both by algorithms, such as the metrics used by the different dispatching rules described in section 3.5.2, and by upper layers to provide feedback for users. The metrics module is responsible for performing the calculations of these metrics. This involves getting the required information from the model, performing the calculations and then storing the derived values at appropriate places in the model.

7.2.5.2 Reports

The Reports module is intended to be used when processing of information is required to facilitate special reports and other visualizations. This can for example be to structure the information to be suitable for use by an interactive Gantt chart and also for providing functionality to communicate back to the model on actions performed in the interactive chart. The report requirements from the case study do not require the use of the report module.

7.2.6 Interface/API

The framework should provide a complete interface for its functionality via the Interface/API. This layer is a façade [13] for the framework. Exactly how this should be designed cannot be determined at this point. It has to be further investigated when the framework is integrated with ComActivity's solution. For the purpose of the prototype it is sufficient to provide functionality to load the model and to run the algorithm.

7.3 The Prototype

The purpose of the prototype is to implement the model and algorithm modules of the scheduling core layer to be used as a proof of concept. Some parts of the Metrics module are also implemented to show how information required by reports and visualizations can be derived.

To test the prototype required parts of the data access, extraction/transformation/loading/updating and interface/API modules have been implemented but the details of this are not described.

7.3.1 Functionality

The functionality provided by the prototype is the possibility to produce schedules by applying the algorithms described in section 7.2.4 to the model described in section 7.2.3. The algorithms described in Figure 7.9 to Figure 7.18 are implemented together with a control interface. A prerequisite is that the model is loaded with data.

7.3.2 Implementation

The prototype consists of four top-level packages model, algorithms, metrics and util. These packages contains sub-packages that further groups the classes. A diagram of the top-level classes is shown in Figure 7.19.

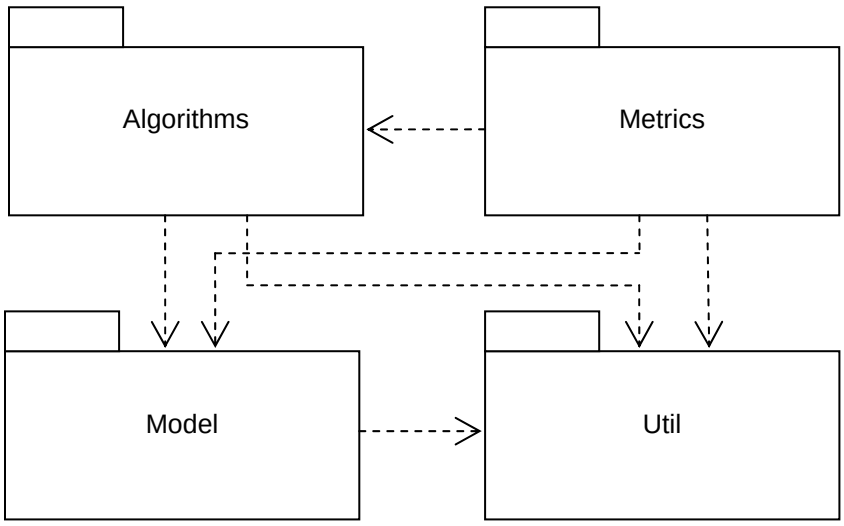


Figure 7.19. UML package diagram of top-level packages of the prototype.

7.3.2.1 Model

The model package consists of the classes that implement the model as described in section 7.2.3. These classes are grouped into appropriate sub-packages as shown in Figure 7.20.

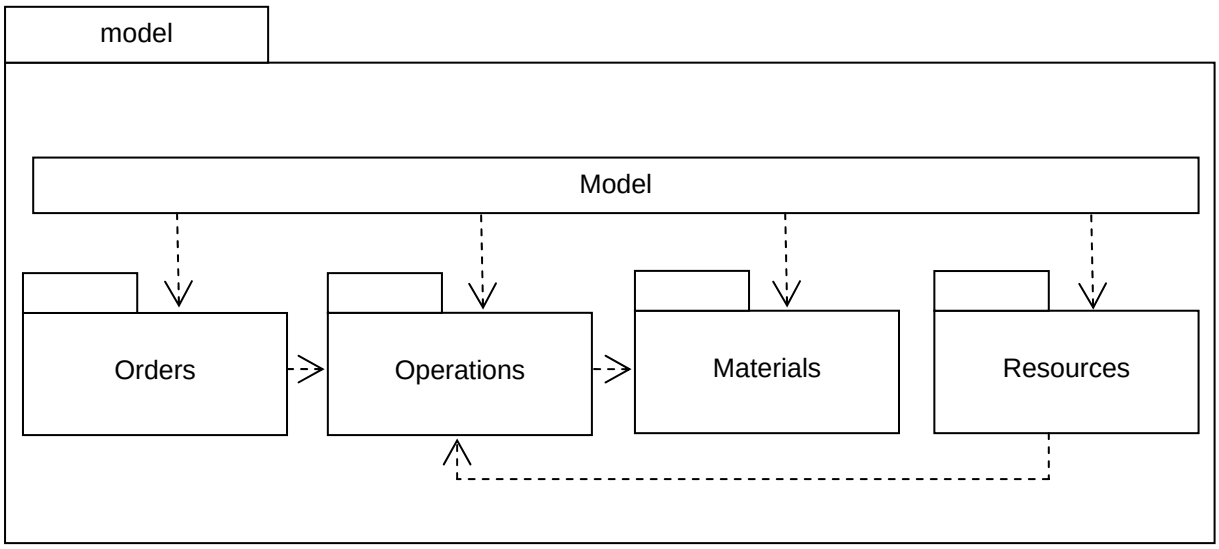


Figure 7.20. The model package.

The class Model is a façade that provides access to the model. The orders sub-package contains the classes shown in Figure 7.7 except the Operation class, the operations sub-package the classes described in Figure 7.6 and the materials sub-package the classes shown in Figure 7.8. The resources sub-package contains the classes shown in Figure 7.3, Figure 7.4, Figure 7.5 and a class representing the calendar as described in section 7.2.3.2.

The data structures of the model are implemented using the Java Collections framework [38][7]. See appendix A for the model package code.

7.3.2.2 Algorithms

The algorithm package contains classes that implement the algorithms and classes that perform calculations and comparisons. Figure 7.21 shows the algorithm package.

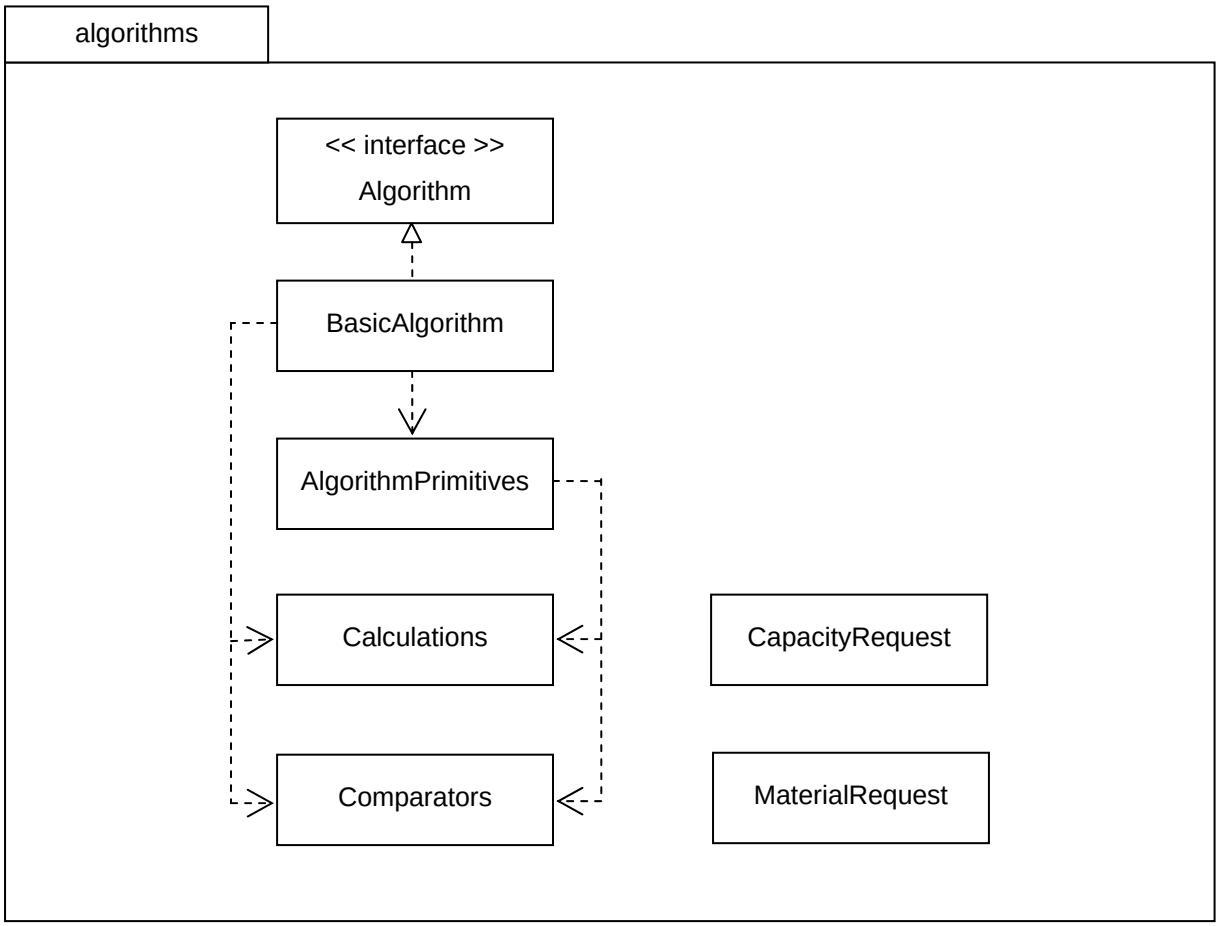


Figure 7.21. The algorithm package.

The algorithm interface should provide the interface for clients to control the algorithm in the package. In the prototype this just contains the method *runAlgorithm*. The *BasicAlgorithm* class implements the algorithm described in section 7.2.4.2 and *AlgorithmPrimitives* the algorithms described in section 7.2.4.1. The *Calculations* class implements calculations and logic that are required by the algorithms and by the metrics package. The *Comparators* class implements comparators that are required for sorting the collections contained in the model package. The *CapacityRequest* and *MaterialRequest* classes are command classes used to transfer parameters in the algorithm classes. See appendix A for the algorithm package code.

7.3.2.3 Metrics

The metrics package contains classes for performing calculations and retrieving of information required in reports and other visual presentations. Figure 7.22 shows a diagram of the package.

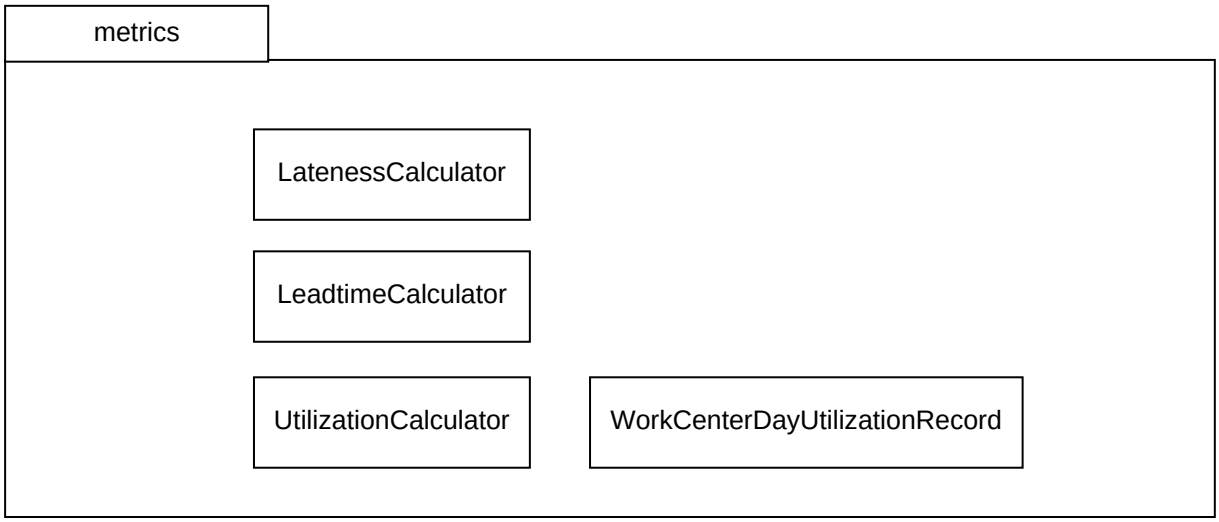


Figure 7.22. The metrics package.

The calculations implemented are lateness of orders and operations, leadtime of orders and utilization of work centers. The classes for this are *LatenessCalculator*, *LeadtimeCalculator* and *UtilizationCalculator*. The *WorkCenterDayUtilizationRecord* class is used by the *UtilizationCalculator* to store information. See appendix A for the metrics package code.

7.3.2.4 Utils

The util package contains utility classes used in the scheduling framework. These are classes to handle the management and manipulation of date and time and for handling constants. Figure 7.23 shows the util package.

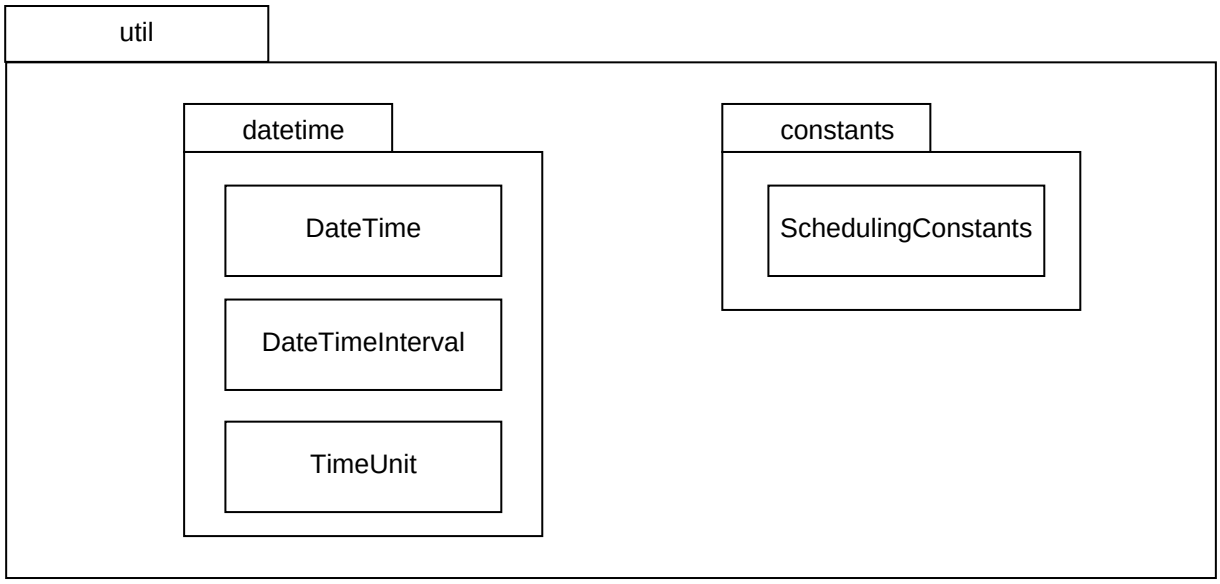


Figure 7.23. The util package.

See appendix A for the util package code.

8 Results

In the preceding chapters, production scheduling and production scheduling systems has been investigated. As described in chapter 2, scheduling is the last step in a chain of activities carried out in a company to perform production. Scheduling determines in detail how production is executed on the shop floor and thus directly affects the overall performance of a company. Therefore scheduling, and how it is performed, is an important part of the management of a company.

There exists a great deal of interest from industry to utilize software systems to support and improve scheduling and to facilitate the application of advanced scheduling methods. The purpose of this dissertation has been to investigate which functional requirements there are on such systems. Since the practical use of advanced scheduling methods has shown to be problematic, the aim has been to find requirements that are derived from real production environments and methods that can be put to use in such environments. The investigation has been conducted through literature studies and by performing a case study in a company that uses a scheduling system. From the requirements derived, a scheduling system framework has been proposed and a prototype of the core functionality in this framework has been implemented.

The case study has been a very important part of the investigation. By using experiences gathered from the practical use of a scheduling system it has been possible to find requirements that are derived from real needs. Although the aim in this dissertation has been to be as general as possible about production scheduling, the size and diversity of the area requires a tighter focus and the case study has provided for this. The type of production environment that exists in the shop studied in the case study has been the basis of the investigation. This environment can be characterized as a job shop with diverse production and small order quantities with often complex relationships between orders.

8.1 Difficulties in the implementation and use of scheduling systems

Since the implementation and use of scheduling systems has shown to be problematic, it has been important to investigate the reasons for this and how these difficulties may be overcome. As described in chapter 3, the difficulties in the implementation and use of a scheduling system can be divided into problems concerning the modeling of production environments and problems involving human aspects of using the system.

8.1.1 Modeling difficulties

To be able to use a software system for scheduling, the production environment must be represented as a model in the system. Due to the fact that a typical production environment is very complex and always involves some degree of uncertainty, developing a model of such an environment can prove difficult. As with all modeling activities the problem is to find a good enough approximation of reality without getting the model to complex. In a scheduling system the model must represent the production environment in enough detail to make the developed schedules useful. What enough detail in the model actually means must be determined for the particular environment in question. Factors that must be considered when determining this are:

- *The purpose of the developed schedules.* As described in chapter 3, schedules can be used for other purposes than the obvious one of how to execute production on the shop floor. For instance, if a schedule is just to determine system capacity for a higher level production planning system, the required level of detail might be less than if the schedule is to be executed in reality.
- *Handling of uncertainties.* The uncertainties that can occur in a production environment are described in section 3.6.1. The general rule related in section 3.5 which says that the more randomness that exists in an environment, the simpler the scheduling methods should be, also applies to the model and is very important to keep in mind. The value of putting effort into detailed modeling of an unstable environment with high levels of randomness is questionable. It is probably wiser to first try to remove as many of the uncertainties as possible. Some level of uncertainty will always exist and this can be handled both in the model and in the applied algorithms. The most common method is probably a combination of the two. In the environment described in the case study uncertainties are handled by setting the utilization rate of

resources to less than 100% and using the slack put into the schedule by this to deal with unforeseen events.

- *Effort required to keep the model updated.* If the model is complex and contains many details, it requires more effort to keep it updated. The cost and time required to keep the model updated must be put in relation to the value of the effort. That is, the value of the schedules produced must be worth the effort required to develop them. Since an updated model is most often as important as a sufficiently detailed model when it comes to developing schedules that are feasible, these two aspects must be considered together.
- *The information requirements of the algorithms that are going to use the model*. Different algorithms can have different information requirements which must be satisfied by the model. In Figure 7.2, a basic model with extensions for different algorithms is shown. In such a design it is important to notice that sufficient level of detail must exist in the basic model, the extension can then add details.
- *General versus special cases*. It is important to differentiate between the general case and cases that occur in special situations. This is related to the effort to keep the model updated as described above. An example of this from the case study is when parts of an operation require two workers to perform. If this should be modeled the operation would have to be split into two (or even three) operations. The effort required to do this has not been termed necessary, instead the time required from the second worker is taken from the slack that is put into the schedule by setting the utilization rate of the resources to less than 100%. In the environment described in the case study, that a part of an operation which requires two workers has been termed a special case. When determining the level of detail to describe in a model, one approach can be to start with the general case and then add the special cases that are shown to occur most frequently and have the most impact on scheduling. Another way of handling special cases is to let the algorithm find them and signal this. It is then up to the scheduler to handle these cases manually in some way.

A general principle that is very important when modeling a production environment is that the complexity of the model and the effort required to keep it updated must always be put in relation to the value of the schedules produced. If for instance the scheduling method proposed to be applied to an environment requires a complex model which in turn requires

much effort to keep updated, maybe a less ambitious scheduling method requiring a simpler model should be considered. The solution to strive for would be the simplest model and scheduling method that satisfies the scheduling requirements, and by that the general goals, of the company.

8.1.2 Human aspects

As described in section 3.6.2, scheduling has traditionally been a manual operation performed by skilled and experienced personnel with thorough knowledge of the production environment in question. A scheduling system can never replace this; it will always be a tool to be used by the person developing the schedules. Scheduling systems have a reputation of being complicated to use and produce schedules that are hard to understand. This has been one cause for failed implementations of such systems. To get a system accepted and seen as an aid in the scheduling activities it is important that the system is easy to use and that the schedules produced by the system are feasible (possible to execute on the shop floor) and transparent (the user should be able to follow how the schedule has been derived).

When implementing a scheduling system it is very important that all parties affected by the system understand and accept the implications of using such a system. Due to the general difficulties involved in the scheduling activities, such as prediction of the future and handling of large amounts of information, when this process is formalized in a system it becomes very important that all interested parties perform the activities required for the system to work. Examples of this are to follow the developed schedules and reporting performed work. When using a scheduling system, all relevant information must be available in the system. It is very easy to make a schedule infeasible by for example taking decisions that are not in accordance with the information in the schedule or to use informal ways, which are not reflected in the system, to exchange information. Such decisions might affect other parts of the schedule, potentially making dependent activities impossible to perform and thus making the schedule useless. Getting interested parties involved is therefore necessarily a large and important part in the implementation of a scheduling system. This is probably also a hard part since it is not uncommon in planning situations that people are used to taking fast decisions to solve problems that occur. When a scheduling system is used, such decisions must fit in with the information that exists in the schedule. By some people in an organization, a scheduling system can therefore be seen as obstacle in their daily work. Getting those people to accept the use of the system, and hopefully beginning to see how the system can support them in their work, is very important. One reason to use a scheduling system is often to get a global

view of the scheduling situation and take decisions based on this view. If people continue to take local decisions the system is not going to work as intended. It is easy to see that this can be a problem since in reality there can exist situations where a fast decision can make it possible to take advantage of an opportunity that occurs, for instance by unexpected available capacity in a production resource. It could be argued that a scheduling system makes it difficult to take advantage of such situations. This is an aspect that must be considered when deciding on the scheduling method to apply in an environment and is connected to how much uncertainty there is in the environment. Opportunities like the one described above occur because of some fact that was not known when the schedule was developed. If this occurs often the scheduling method might not be suitable for the environment. Since all environments contain some level of uncertainty, one approach could be to allow local decisions to some degree. In the environment described in the case study, this is accomplished by setting a lower utilization level for resources. The slack put into the schedule by this leaves room for local decisions. The difficult part is to know when the local decisions affect the schedule so much that a reschedule is required. In the job shop studied no explicit rule for this exists, but since a reschedule is performed every day the effects of the local decisions are incorporated daily into the schedule.

Getting interested parties involved in the implementation of a scheduling system and gaining a wide acceptance is crucial for a successful implementation. The possibilities available in the scheduling system to be adapted to a particular environment largely affect this. The system must be flexible enough to provide the scheduling functionality required and provide user interaction that suits the environment. Apart from possibility to support the scheduling method decided for the production environment, without which the system hopefully would not have been chosen, the ability to provide suitable user interaction is very important for the acceptance of the system. This is of course a general requirement for software systems, but it becomes even more important here because of the circumstances described above where there can exist arguments for not using the system. If the user interface can be highly adapted to the tasks to be performed, it is easier to get people involved and hopefully realize the support the system can provide.

8.2 Reasons for implementing a scheduling system

The overall reason for using a scheduling system is of course to produce schedules that satisfy the scheduling objectives of the company (see section 3.2). It is useful to investigate this in more detail and determine what the scheduling system can provide that is hard to perform in a manual system. As described in chapter 3, there can be different reasons for developing a schedule. The most common reason and the one discussed here is to develop a schedule that can be executed on the shop floor. From the environment described in the case study, the following reasons where termed important when the decision to implement a scheduling system was taken (see section 6.1.1):

- Obtain a global view of the current production situation that is shared among all participants in the scheduling process.
- Take informed scheduling decisions that satisfy the overall objectives of the shop and consequently minimize local, potentially sub-optimizing, decisions.
- Execute, and provide feedback on, the schedule decided on the shop floor
- Reduce lead-time and improve the accuracy in the due dates that are promised to customers
- Solve problems early, hopefully before they disturb the production and affects deliveries.

These reasons are probably fairly general and occur in most production environments where the implementation of a scheduling system is considered. The aspect that appears to be most important is the visualization of the production situation. To share a common view of the current production situation, and to use this as a base for scheduling decisions, is an important reason to use a scheduling system. To visualize the production situation, and to communicate this, appears to be more important than for the system to produce an optimized schedule. It can be argued that to have a common view of the production situation among the interested parties is a requirement before advanced scheduling methods can be applied. Without this common view, even if an optimal schedule existed, it would be difficult to use in practice since it could not be communicated in the required way. The human aspects of this were described in section 8.1.2.

To produce this common global view of the production situation requires a significant amount of information processing that a scheduling system can provide. The level of detail required in this global view, and the cost of keeping the information updated, in relation to the value the view provides where discussed in section 8.1.1.

8.3 Application of advanced scheduling methods and theories

From the discussion in sections 8.1 and 8.2, it can be argued that the most important part of a scheduling system is the ability to provide a common global view of the production situation in an environment and to visualize this in different ways. Only when this is in place it is possible to consider the application of advanced scheduling methods. As described in chapter 3, the use of optimizing methods is only in question when the production environment is very stable and the level of randomness and uncertainty are low. The use of optimizing methods also probably incurs higher cost to keep the model updated since the requirement of details in this is high. For most production environments, the question is to find a suitable scheduling method that can work with the uncertainties in the environment while at the same time giving the possibilities to fulfill the scheduling objectives of the company. From the requirements derived from this it can be useful to study the theoretical aspects of scheduling methods which were overviewed in chapter 4. The application of these theories must always be with a model that is possible to keep updated as described in section 8.1.1 in mind, together with the requirement from section 8.1.2 that the developed schedules must be transparent and possible to understand. The use of a complex method might require specialized skills to understand and control from the personnel using it.

One approach to the application of advanced scheduling methods might be to start with the common global view as described above and then apply more advanced method as the need occurs. When this global view is in place and all interested parties shares a common view the application of more sophisticated methods might come naturally. Information can probably be derived from this visualization of the production environment that has not been available before, and this might reveal possibilities to apply scheduling methods that was not apparent earlier.

8.4 Requirements on a scheduling system

Section 7.1 describes requirements on a scheduling system gathered from literature studies as related in chapters 2 to 5 and the case study described in chapter 6. These can be grouped into external requirements concerning the usage of the system and those of a more technical, internal character. Many of the external requirements concern the possibilities of visualizing the production environment in a way that suits the particular needs of the environment in question and the ease of use of the system. The internal requirements are derived from the external requirements and must be satisfied to provide the functionality needed to satisfy the external requirements.

8.4.1 External requirements

The external requirements can in large part be derived from the discussion in section 8.1.2 on the human aspects of implementing a scheduling system and the discussion in section 8.2 on the reasons for implementing a scheduling system. External requirements can be grouped into visualization of the production environment and control of the developed schedules.

8.4.1.1 Visualization

Arguably the most important aspect is to provide a visualization of the production environment. The reason for this is that the visualization is what makes it possible to get a common global view of the production environment that can be shared among all parties affected by the schedule. This is fundamental to the successful use of a scheduling system.

Visualization is accomplished via reports and other graphical representations. Examples of this are Gantt charts and load diagrams as described in section 5.2.1.3. Since how this is done plays a crucial part in getting a scheduling system accepted and actually used, it is important that the system provides flexible means for this.

8.4.1.2 Control

To control how the schedule is developed is also an important part in the external requirements on a scheduling system but since this affects fewer users these requirements are not as important as the visualization requirements. This said, there exists no reason why a scheduling system cannot satisfy both requirement groups equally well.

The control of the development of schedules involves such aspects as setting parameters that affects how schedules are developed and also encompasses the requirement that the schedules produced should be possible to understand. It is important that the system provides the scheduling support required for the environment in question and should be possible to

adapt to the particular needs of the users. Examples of this can be extensions of the requirements for visualization such as interactive Gantt charts where the schedule can be manipulated.

8.4.2 Internal requirements

The internal requirements are derived from the external requirements and are requirements on the core functionality of the system. Examples of this are interaction with different data sources, flexible and adaptable modeling of production environments and possibilities to implement and apply different scheduling algorithms.

The most important part is the modeling capabilities of the system. It must be possible to develop a model that represents the production environment in a way that is suitable for the chosen scheduling method. The model must provide information required for the scheduling algorithms that are applied as well as for the different reports and graphical representations that are used to visualize the production environment. With the model as a base, the system must then provide ways of implementing the algorithms required for the chosen scheduling method.

8.4.3 A proposed scheduling system framework

The proposed scheduling system framework is described in chapter 7. The aim with the framework is to primarily satisfy the internal requirements as described above. The core functionality of the system determines how the external requirements can be satisfied and the design of this is the most important. To satisfy the external requirements, the main aspect is to make sure that all information that is needed is actually available. If the required information is available, the visualization of the production environment becomes presentation issues, that of course are very important, but there exists well established techniques to solve these issues. If the information exists it is be possible to present it in the way required by the users of the system. Therefore the main focus of the framework is the core functionality.

An overview of the framework is shown in Figure 7.1, and section 7.2 describes the design. The aim of the design has been to be as general as possible and to keep the framework modular to make it adaptable to different environments. The detailed parts of the framework, where it has been difficult to have a general approach, has been design to satisfy the requirements of the environments of the case study. The model component of the framework has been design to be as independent of the underlying data sources as possible to provide for use of different underlying environments. The basic model can also be extended to satisfy

different requirements from applied algorithms as shown in Figure 7.2. The interface against clients is thorough a façade that should give access to all functionality provided by the framework. The framework includes separate modules for reports and metrics in which processing can occur to provide information required in reports and other graphical representations.

9 Evaluation of proposed scheduling system framework

A prototype that includes the core functionality of the framework described in chapter 7 has been implemented. The intention of the prototype has been to determine how the proposed framework can be used to support scheduling in the environment described in the case study (see chapter 6). The evaluation has been made by validating and comparing the schedules produced by the prototype against schedules produced by the existing scheduling system and by evaluating how the reporting and visualization needs are satisfied by the framework.

9.1 Validation and comparison of schedules

The algorithm implemented in the prototype is similar to that utilized in the scheduling system in use in the environment described in the case study. It should therefore be possible to produce schedules from a determined set of production data and then compare the schedules produced by the two systems from this data. When examining the schedules produced in detail, it has been shown that the schedule produced by the existing system contains faults. Operations are actually scheduled using more capacity than has been described as available. Since the deviation from what is the correct capacity is small the fault has gone unnoticed and it probably has not affected the performance of the environment too much. Because of this fault, a side by side comparison of the schedules has not been possible. The comparison has instead been performed by taking smaller excerpts of the schedules, such as individual orders, and manually comparing and validating these.

Since the priority rule used is strictly based on the due dates of orders, the validation of this is straightforward and by manually checking the derived dates of selected orders and operations it has been validated that the calculations performed are correct. From this it has been concluded that the schedules produced are valid. The validations have shown that the prototype produces schedules that correspond to the intended outcome of the algorithm used.

That the schedules produced by the existing system contain faults shows the importance of validating schedules produced by scheduling algorithms. This is also another reason for the requirement that how schedules are developed should be transparent, so that they are possible to understand and validate.

9.2 Evaluation of reports and visualization possibilities

The requirements from the environment in the case study concerning reports and visualizations are described in section 6.4. The requirement on the prototype is to show how the information for this is obtained in the scheduling core of the framework. Since these requirements were known when the framework was design all information needs have been built into the framework as described in chapter 7. The intended use of the framework is in the environment provided by ComActivity. The presentation layer of this environment is completely web based and provides all functionality necessary to satisfy the visualization need from the case study. As an example there exists a web based interactive Gantt chart that can be used to display scheduling information.

9.3 Summary

As a summary of the evaluation of the proposed framework it can be concluded that the framework can satisfy the scheduling needs of the environment described in the case study. The schedules produced by the prototype are validated and, since the schedules produced by the existing system contain faults, actually can be described as better than the schedules currently in use. By using the reporting and visualization possibilities provided by the combination of the information produced by the scheduling core of the framework and ComActivity's environment, all requirements described in the case study can be satisfied.

In addition to this there are some advantages that come from the web based deployment such as easier administration and the potential to make the reports and other visualizations available to everyone that needs them. The existing system requires individual installations on every computer where it is to be used.

10 Conclusion and future work

After this investigation into production scheduling and scheduling systems, some conclusions can be summarized and some areas of future work can be suggested.

10.1 Conclusion

The results and conclusions of this dissertation are based on literature studies and the case study as described in the previous chapters. Since the area of production scheduling is so large and diverse it has been necessary to focus on selected parts to maintain a manageable scope. The focus has been kept by using a production environment described in a case study as a reference and by focusing on the aspects most relevant to this environment. This environment is a job shop performing diverse production working with small order quantities and often complex relationships between orders.

The implementation and use of production scheduling systems in practice has been shown to be difficult and there exist many examples of failed implementations. The causes of this can be grouped into either those concerning the difficulties involved in the modeling of production environments or those concerning the human aspects of using scheduling systems.

The difficulties concerning the modeling of production environments are largely due to the complexities and uncertainties contained in the environments. Connected with the modeling difficulties are the gap between theory and practice in the research area which makes practical application of the theoretical methods difficult. The human aspects encompass such areas as the acceptance and trust of the system and whether the system is actually seen as a support for the scheduling activities. The two causes are related. The model of the production environment is the basis of the system and if this does not describe the actual environment well enough, it is not going to produce the intended output and is not going to provide adequate support for the scheduling activities. Such a system is probably not going to be accepted by the users

A conclusion that can be drawn is that the correspondence between the type of environment, how well the model represents this environment and the scheduling method used is very important. The general rule is that the more randomness and uncertainties that exist in an environment, the simpler the scheduling method should be. When determining which scheduling method that should be applied in a given environment, the goals of the

company, the characteristics of the environment and the modeling possibilities must be considered. It is possible that many failed implementations are caused by trying to apply too advanced scheduling methods to environments, actually to models representing environments, which did not have the characteristics suitable for the applied methods. Under such conditions, a system will probably not give sufficient support for the scheduling activities as they are performed in practice, and thus would not be accepted by the users.

From the description above, the most important aspects to consider regarding production scheduling and the use of production scheduling systems that has been derived in this investigation can be stated as follows:

- *A common view of the production situation shared between all parties participating in the scheduling activity is necessary*.
- *A suitable visualization of the production environment is fundamental*.
- *The scheduling method must be suitable for the production environment and for the model of the production environment that is used*.
- *Optimizing scheduling methods are rarely applicable in practice*.

To successfully schedule a production environment it is necessary that all parties participating in the scheduling activities shares a common view of the production situation. The production scheduling activities generally requires the collaboration between a large number of people, all of which are required to perform actions and make decision that concerns the schedule. These actions and decisions must be in accordance with the common view of the production situation. To arrive at this common view, the visualization of the production environment is fundamental.

The scheduling method to apply in a production environment must be chosen to suit both the actual production environment and the model of the production environment that is used. Theses two aspects are of course related but it can, for example, be the case that a scheduling method is found suitable for an environment but the model necessary for this method requires too much effort to keep updated.

It can also be concluded that the use of optimizing scheduling methods in practice is rarely necessary. This is due to the dynamic character of the production environments considered in this investigation and the uncertainties that will always be a part of such environments.

A viable approach concluded from this investigation is to first focus on modeling and visualization of the environment in question and from that apply scheduling methods as is found suitable. A prerequisite step for this must of course be to determine the scheduling requirements of the environment which are to be derived from the overall goals of the company. The scheduling framework proposed in this dissertation is intended to provide support for applying this approach. The combination of the framework and the possibilities of the web technologies provided by ComActivity's environment give flexible modeling and visualization capabilities and support for the implementation and application of scheduling algorithms. The framework has been designed to be as flexible and transparent as possible and to be adaptable to different scheduling needs.

As has been described in this dissertation, implementing a scheduling approach in a production environment involves many aspects and can therefore be a very complex project which often requires the participation of a large number of people from different departments in the company. An important consideration in such an implementation project is to focus on the implementation of a scheduling approach, not the implementation of a scheduling software system. The software system should, of course, be adapted to the environment not the other way around. This requires a system that is flexible enough to provide this functionality. The implementation of a scheduling approach often requires changes in the environment and the task performed by the personnel involved but it is important that the changes are not dictated by the software system.

It is hoped that the proposed scheduling system framework can be used to support the scheduling activities in real production environments. The framework should be flexible enough to be possible to utilize for different scheduling needs. Important aspects of the framework are the modeling and visualization possibilities. By utilizing the web based architecture provided by ComActivity the deployment and administration of the system should be easy. The approach described in combination with a flexible scheduling system should give the opportunity to utilize a stepwise, iterative development approach where the applied scheduling method can be developed and refined as experience is acquired and knowledge elicited from using the visualization possibilities provided by the framework. This gives opportunities for implementations of scheduling systems that make less impact on the production environment in question and which also should be less costly than other implementation methods.

10.2 Future work

The most interesting, and also most important, future work should be to test the proposed scheduling framework in a real production environment. The aim throughout this investigation has been to maintain a practical focus and the best way to continue would be to use the framework to implement a scheduling system in a company. To do this, the presentation layer of such a system must be implemented. The information required in the presentation layer is present in the framework but it remains to design the actual report and other graphical representations. This should of course be done in collaboration with the intended users.

Another area that should be investigated further is scheduling algorithms. The algorithm that is implemented in the prototype is rather basic and should be possible to improve. One aspect to investigate further is for instance the handling of uncertainty. In the implemented algorithm, uncertainty is handled by decreasing the resource utilization rate. It should be useful to find alternative methods to this since some amount of uncertainty will always remain in all production environments and this must be handled in some way. Different methods would provide for increased flexibility and give more alternatives when satisfying different scheduling requirements. It should also be interesting to more thoroughly study the advanced algorithms overviewed in chapter 4 and investigate how they could actually be used in reality and implemented in the proposed framework.

References

[1] Adams, Balas, Zawack. *The shifting bottleneck procedure for job shop scheduling.* Management science, vol. 34, no. 3. 1988.

[2] Allen. *Maintaining Knowledge about Temporal Intervals*. Communications of the ACM, 26(11), 832 – 843. 1983.

[3] APICS, The Educational Society for Resource Management. www.apics.org.

[4] Aytug, Lawley, McKay, Mohan, Uzsoy. *Executing production schedules in the face of uncertainties: A review and some future directions.* To appear in European Journal of Operational research. (vol. 161, no. 1, 86-110, 2005).

[5] Balzewicz, Domschke, Pesch. *The job shop scheduling problem: Conventional and new solution techniques.* European journal of operational research, 93, 1 – 33. 1996.

[6] Blackstone. *Theory of constraints – a status report*. International journal of production research, vol. 39, no. 6, 1053 – 1080. 2001.

[7] Bloch. *Effective Java*. Addison-Wesley. 2001.

[8] Chun. *Constraint Programming in Java with JSolver.* Proceedings of the Practical Appplication of Constraint Technologies and Logic Programming. London. 1999.

[9] ComActivity. www.comactivity.net.

[10] Cox, Blackstone. *The APICS Dictionary.* APICS. 10th edition, 2002.

[11] Fowler. *UML Distilled*. 3rd edition. Addison-Wesley. 2004.

[12] Fox, Sadeh. *Why Is Scheduling Difficult? A CSP Perspective.* Center for Integrated Manufacturing Decision Systems. Carnegie Mellon University. 1990.

[13] Gamma, Helm, Johnson, Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley. 1995.

[14] Garey, Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company. 1979.

[15] Goldratt, Cox. *The Goal: A process for ongoing improvement.* North River press, 2th edition. 1992.

[16] Higgins. *Architecture and Interface Aspects of Scheduling Decision Support.* Human Performance in Planning and Scheduling. Taylor and Francis. 2001.

[17] J2EE, Java Enterprise Edition. www.sun.com.

[18] Jain, Meeran. *Deterministic job-shop scheduling: Past, Present and future.* European journal of operational research, vol. 113, no. 2, 390 – 434. 1999.

[19] Kempf, Uzoy, Smith, Gary. *Evaluation and comparison of production schedules.* Computers in industry. No 42. pp 203 – 220. 2000.

[20] Le Pape. *Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling systems.* Intelligent Systems Engineering. 3. 55 – 66. 1994.

[21] Marriott, Stuckey. *Programming with Constraints: An Introduction*. MIT Press. 1998.

[22] McKay, Safayeni, Buzacott. *Job-Shop Scheduling Theory: What is relevant?* Interfaces, 18:4, 84 – 90. 1988.

[23] Michalewicz, Fogel. *How to solve it: Modern Heuristics.* Springer-Verlag. 2002.

[24] Nahmias. *Production and Operations Analysis.* McGraw-Hill. 4th edition, 2001

[25] Pinedo. Scheduling. *Theory, Algorithms and Systems*. Prentice-Hall. 2nd edition, 2002.

[26] Plenert, Kirchmier. *Finite capacity scheduling*. John Wiley & Sons. 2000.

[27] Pongcharoen, Hicks, Braiden. *The development of genetic algorithms for the finite capacity scheduling of complex products, with multiple levels of product structure*. European journal of operational research, 152, 215 – 225. 2004.

[28] Porter, Little, Peck, Rollins. *Manufacturing classification: relationships with production control systems*. Integrated Manufacturing Systems. 10/4, 189 – 198. 1999.

[29] Portougal, Robb. *Production scheduling theory: just where is it applicable.* Interfaces, 30:6, 64 – 76. 2000.

[30] Ralston. *A brief history of manufacturing control systems, part 1*. Control, June 1996.

[31] Ralston. *A brief history of manufacturing control systems, part 2*. Control, July/August 1996.

[32] Sadeh. *Micro-opportunistic scheduling: The Micro-Boss factory scheduler.* Technical report CMU-RI_TR-94-04. The Robotic Institute, Carnegie Mellon University. 1994.

[33] Stoop. *The complexity of scheduling in practice.* International journal of operations and production management, vol. 16, no. 10, 37 – 53. 1996.

[34] Uzoy, Wand. *Performance of decomposition methods for job shop scheduling problems with bottleneck machines.* International journal of production research, vol. 38, no. 6, 1271 – 1286. 2000.

[35] Vollman, Berry, Whybark. *Manufacturing planning and control systems*. McGraw-Hill, 4th edition, 1997.

[36] Wiers. *A review of the applicability of OR and AI scheduling techniques in practice.* International journal of management science, 25(2), 145 – 153. 1997.

[37] Yen, Chow, Yau. *On the design and Development of User Interfaces in Interactive Scheduling Systems*. Manufacturing Agility and Hybrid Automation. 1998.

[38] Zukowski. *Java Collections*. Apress. 2001.

A Prototype code

The code for the prototype together with corresponding javadoc documentation is contained in a jar file called *scheduling-framework.jar*. The following packages and source files are included:

algorithms

- Algorithm.java
- AlgorithmPrimitives.java
- BasicAlgorithm.java
- Calculations.java
- CapacityRequest.java
- Comparators.java
- MaterialRequest.java

metrics

- LatenessCalculator.java
- LeadtimeCalculator.java
- UtilizationCalculator.java

model

- Model.java

model.materials

- Material.java
- MaterialDelivery.java
- MaterialRequirement.java
- Materials.java
- MaterialShortageRecord.java

model.operation

- Operation.java
- OperationMaterialRequirement.java
- Operations.java

model.orders

- CustomerOrder.java
- CustomerOrderLine.java
- CustomerOrders.java
- ProductionOrder.java
- ProductionOrders.java

model.resources.calendar

Calendar.java

model.resources.capacityadjustments

CapacityAdjustment.java

CapacityAdjustments.java

model.resources.shifts

Shift.java

ShiftDay.java

ShiftDayPart.java

Shifts.java

model.resources.workcenters

TimePerDayRecord.java

UtilizationRecord.java

WorkCenter.java

WorkCenterResource.java

WorkCenters.java

util.constants

SchedulingConstants.java

util.datetime

DateTime.java

DateTimeInterval.java

TimeUnit.java