Guidelines for the deployment and implementation of manufacturing scheduling systems

Jose M. Framinan[1], Rubén Ruiz[2*]

[1] Industrial Management, School of Engineering, University of Seville,
Ave. Descubrimientos s/n, E41092 Seville, Spain, jose@esi.us.es

[2] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática,
Universidad Politécnica de Valencia, Valencia,
Camino de Vera s/n, 46021 Valencia, Spain. rruiz@eio.upv.es

March 26, 2010

Abstract

It has been frequently stated that there exists a gap between production scheduling theory and practice. In order to put the theoretical findings into practice, the advances in scheduling models and solution procedures should be embedded into a piece of software –a scheduling system– at work in the companies. This results in a process which entails 1) determining its functional features, and 2) adopting a successful strategy for its development and deployment. In this paper we address the latter question and review the related literature in order to identify descriptions and recommendations of the main aspects to be taken into account when developing such system. These issues are then discussed and classified, resulting in a set of guidelines that can help practitioners during the process of developing and deploying a scheduling system. In addition, the identification of these issues can provide some insights to drive the theoretical scheduling research towards those topics more demanded by practitioners, thus helping in closing the aforementioned gap.

Keywords: Deployment, Implementation, Scheduling systems, Architecture.

*Corresponding author. Tel.: +34-96-387-70-07x74946; fax: +34-96-387-74-99.

1 Introduction

Among the scheduling community, the so-called 'gap' between production scheduling theory and practice has been a recurrent theme in many reviews of the field in the last decades, as can be seen e.g. in Graves, 1981; Dudek et al., 1992; MacCarthy and Liu, 1993 and McKay et al., 2002). While it is likely that the reasons for this gap cannot be analysed in isolation from the rest of the quantitative research within the production management field (see for instance the surveys on the usage of operations research techniques in Ledbetter and Cox, 1977, Ford et al., 1987, McKay et al., 1988b, and Olhager and Rapp, 1995), it is true that a small percentage of the scheduling research deals with realistic production settings (Reisman et al., 1997).

In an earlier paper (Framinan and Ruiz, 2010) we identified that, in order to close this gap between scheduling models and procedures, and their implementation in a real manufacturing setting, the former should be translated into a piece of software with a number of functions to support scheduling decisions in a company. This implies carrying out a *software development process* to obtain a final *product*, i.e., a scheduling tool at work. In such software development process, there are a number of technical, human and organisational issues which are critical and should be adequately managed to ensure a successful result. Particularly, two aspects are regarded of utmost importance in this process, i.e.:

- What is the product that should be expected at the end of the process?, and
- How to deploy and implement this product into an specific company?

The first of these two issues relates to identify the business (manufacturing) functions to be covered by the software tool, and it was analised in Framinan and Ruiz (2010). In this paper, we focused onto the main components –software modules– that should constitute the building blocks of a scheduling tool and on their relationship. What it was produced was a so-called *architecture* of a scheduling tool derived from a systematic analysis of the requirements for a scheduling system posed in the related literature. The architecture described occupies an intermediate space between a description of the decision problems to be solved by the schedulers, and the design and implementation of a specific scheduling tool. Therefore, the architecture can influence the system's subsequent detailed design and implementation and –since most parts of this architecture are common to many manufacturing scheduling systems– relying on an effective, validated architecture helps reducing the usually costly and time-consuming software development process while ensuring the quality of the resulting scheduling tool.

In this paper, we address the second issue, i.e., how to bring a scheduling system (designed, or not, around the ideas of the architecture previously mentioned) to a company. This may be certainly be regarded as the "last mile" to be covered, but it is crucial for the success of a

scheduling system, and it is likely to be the cause for some failures. It is therefore, a critical issue that, unfortunately, has been seldomly addressed in the scheduling literature. To the best of our knowledge, we are only aware of the works by Wiers and Van Der Schaaf (1997) and Wiers (2001) introducing design aspects for scheduling decision support systems, i.e.: level of support, transparency, autonomy, and information presentation. However, on one hand the description of theses aspects takes place at a very high level, and on the other hand, these aspects merge issues related to the functionalities of the system (such as the level of support), and guidelines for their design.

While it is obvious that the process of successfully developing and deploying scheduling systems in companies include many enterprise-specific aspects, there are a number of lessons that may be regarded as common for most scheduling systems and that can be thus extracted from an analysis of past implementations. Therefore, the goal of this paper is to discuss and classify a number of evidences supported by previous experiences that can be used as guidelines to effectively conduct the deployment of manufacturing scheduling systems and consequently, to help bridging the gap between scheduling theory and practice.

The remainder of the paper is as follows: in Section 2 we provide a description of the methodology that we have adopted to extract the guidelines, and how we have reviewed and classified the different contributions in the topic. Section 3 is devoted to providing a detailed (one-by-one) discussion of the different guidelines and recommendations. Finally, in Section 4 we draw some conclusions and point out future research lines.

2 Background

The development of a scheduling system is a particular case of that of a Decision Support System (DSS), which in a broad sense encompasses those systems labelled as 'expert systems" (even if in the scheduling field it has been stated that no implementations of these systems has been found in practice, Wiers, 1997). From this viewpoint, software development models employed for business information systems could be applied. In principle, none of these has to be preferred as compared to the others, but given the technical complexity of these systems and the critical nature of the interface with the human –the decision maker–, software development models intensive on prototiping and on fast iterations over versions of the software seem to be more suitable (see Meador and Ness, 1974 for a discussion on the differences between the DSS development cycle and that of other business information systems). In addition, the relatively small size of the scheduling system (at least as compared to other business information system) would not favour the employment of heavily documented and rigid approaches that may increase the burden and costs of software development. We will go through this idea later on when elaborating the guidelines.

In spite of the above, it is possible to elaborate a number of recommendations or guidelines for developing scheduling systems, given the specific nature of the scheduling function. The aim is to provide stakeholders with a set of policies that may help them in the developing process. These policies are based on an analysis of the existing literature describing development cases of scheduling systems, on the requirements established by the scheduling architecture proposed in Framinan and Ruiz (2010), and on our hands-on experience. These guidelines have been grouped into specific topics in an attempt to provide a more structured vision of the recommendations, but we believe that a further systematization of these guidelines is company- and technology-dependent, and thus would not be applicable in many cases. Therefore, there is no intention of being comprehensive but informative and still, some of the hints (even if supported by literature and/or experience) may be debatable for particular developments.The different aspects that would be taken into account are the following:

- **General hints**. Under this umbrella we describe a number of hints (we do not dare to call them "principles") that can be useful to drive the deployment effort. They relate to each other only in that they may potentially affect to all stages in the deployment process.
- **Data**. Here the main issues related to data acquisition, manipulation, and maintenance are to be discussed.
- **Objectives and constraints**. These guidelines refer to selecting and prioritising the objectives and constraints to be incorporated in the scheduling tool. Given the –in practice– close connection between constraints and objectives, we have decided to group both aspects.
- **Solution procedures**. These guidelines refer to how to approach the design and implementation of the solution procedures included in the scheduling system.

Table 1 summarises these different aspects, together with the main references that have been employed to derive these guidelines. Their detailed discussion will follow in the next section.

Type	Guideline	Main references
General hints	Incremental deployment of the scheduling system	Leachman et al. (1996); Wiers (2002); McKay and Black (2007); Missbauer et al. (2009)
	A-B-C Analysis	Bensana et al. (1986); Bitran and Tirupati (1988); McKay et al. (1988b); Kerr (1992); MacCarthy and Wilson (2001)
	Avoid technology-dominant approaches	Laforge and Craighead (2000); Missbauer et al. (2009)
	Modular design and implementation	Kathawala and Allen (1993); Higgins (1996)
	Keep layout modelling simple	Pinedo and Yen (1997); Benavides and Carlos Prado (2002); Freed et al. (2007)
Data	Development of a database interface	McKay et al. (1988b)
	Data quality	Leachman et al. (1996); MacCarthy and Wilson (2001); Berglund and Karltun (2007)
	Data abstraction	McKay and Wiers (2001)
	Keeping interfaces simple	Wiers (2002); Missbauer et al. (2009)
Objectives and Constraints	Prioritisation of objectives	Buxey (1989); Bhattacharyya and Koehler (1998))
	Postpone the selection of objectives	Cowling (2003); Gao and Tang (2008); Graves (1981)
	Objectives versus constraints	Bhattacharyya and Koehler (1998); McKay and Wiers (2001)
	"New" types of constraints	Fox and Smith (1984); Grant (1986); Crawford and Wiers (2001)
Solution procedures	Focus on feasibility	Benavides and Carlos Prado (2002); Cowling (2003); Graves (1981)
	Taking into account the available decision time	Cowling (2003)
	Providing a set of alternative solutions	Tang and Wang (2008)
	Transparency of solution procedures	Wiers (1997); Baek et al. (1999); Laforge and Craighead (2000); Wiers (2001); Cowling (2003); Freed et al. (2007)
	Design of algorithms	Wiers (2001)
	Manipulation of solutions	Collinot et al. (1988); Bitran and Tirupati (1988); Numao and Morishita (1989); Prietula et al. (1994); Sauer and Bruns (1997); T'kindt et al. (2005); Pinedo (2007)

Table 1: Main contributions from which the guidelines have been extracted.

3 Guidelines

In this section we provide a detailed explanation and justification of each one of the guidelines summarised earlier in the paper. The classification of the guidelines employed there is also used here to classify the different recommendations. When appropriate, references to case studies and/or scheduling systems' development is given.

3.1 General hints

3.1.1 Incremental deployment of the scheduling system

As just any other business information system, scheduling systems are more suitable to be implemented in companies by employing an incremental approach where progress can be rapidly visualised rather than "big-bang" approaches, where a final solution is envisioned and then implemented. Despite their potential shortcomings –being the most notable the increase in the deployment period–, the literature on implementation of business information system is clearly stressing the suitability of an incremental approach and particularly when these information systems contain company-specific features, which is precisely the case for the scheduling function. Indeed, several cases report the need of long, iterative approaches for the successful implementation of scheduling systems (see e.g. Leachman et al., 1996; Wiers, 2002) as opposed to the classical waterfall approach. This rationale is also related to the best fit of a bottom-up approach (typical in incremental approaches) as compared to a top-down approach (typical for big-bang approaches). Therefore, even if the final goal would be to capture with maximum detail the complexity of the physical shop floor, a more simple model would suffice for launching the scheduling tool, and for making the first results of its adoption visible, which in turn would serve to ease the subsequent deployment steps (Missbauer et al., 2009). Note that some authors state that simply a visual representation of the Gantt schedules is sometimes regarded as a big advance with respect to the previous situation (see e.g. McKay and Black, 2007) and thus can be used to visibilize the advantages of a scheduling software.

3.1.2 A-B-C analysis

Scheduling is about making and monitoring plans, not about capturing and responding to all possible exceptions and alternative scenarios, which are virtually infinite. There is a big temptation in stretching the limits of a tool which is potentially able to make plans with great detail. But with the ever increasing number of product variants manufactured by companies due to the rise of mass customisation, nearly all real-life shop floors would be open shops with very complex constraints in strict terms. However, the effort to model this complex shop floor has to be carefully weighted: For instance, it may turn out that, for most product variants, this complex setting results in a much simpler layout. Indeed, it is often claimed that many jobshops are flowshops

for most of the jobs (see e.g. Storer et al., 1992; Knolmayer et al., 2002). Similar simplifications can be achieved by concentrating the scheduling activities on a few key operations (MacCarthy and Wilson, 2001). On the other hand, in many factories producing a standard set of products there is a percentage of jobs which are customised. Another source of complexity in a scheduling system comes from re-working some jobs due to quality defects. Unless the percentage of quality defects or the percentage of customised jobs is really high, it may be easy to implement simple shop floor directives, such as giving the highest priority to these jobs, so to avoid a more complex approach when modelling the shop.

There are additional reasons to perform such an attempt to simplify the resulting models, as the literature describing cases consistently states that not all details of the shop floor can be captured into the scheduling system (see e.g. Bensana et al., 1986; McKay et al., 1988a). Indeed, in Kerr (1992) it is described that the attempt to carefully incorporate all constraints observed by the human scheduler was identified as one of the roots of the failure of the system.

Therefore, an instrument to obtain these reductions in the complexity of the original shop floor is a Pareto (or A-B-C) analysis based either in the contribution of the jobs to the bottom line, the frequency of the different orders, or the usage of the manufacturing resources (see a similar idea behind the observations in Bitran and Tirupati, 1988 and Webster, 2001). As a result, it seems sensible to develop a system to schedule "just" 80% of the jobs rather than considering a much more complex modelling approach in order to handle the other remaining 20%, particularly if we adopt the aforementioned incremental approach.

3.1.3 Avoid technology-dominant approaches

Whenever possible, the technicalities of the scheduling models and tools should be hidden to the schedulers. Given the complex nature of the scheduling business function, it is unavoidable that the logic of the scheduling system is not understood by many users, and regarded as a black box (Laforge and Craighead, 2000). On the other hand, IT specialists cannot fully grasp the dynamics of the company, so they need the input from these users, a factor which is believed to be a key for a successful deployment (Missbauer et al., 2009). Indeed, it is cited that any technological support for the scheduling function will be successful if due consideration is given to organisational design (Vernon, 2001).

Better training may be needed in the implementation of scheduling packages, as they contain rather complex features and options for configuration (Laforge and Craighead, 2000). Obviously, even if we will recommend to keep these options at a minimum, training must be an important item in the deployment process. Note that this not only would affect to the quantity of the training (as measured in hours), but to the quality of the training. More than often, the user

training on business information systems consists merely on showing which screens and buttons should be pressed to replace the old manual functions of the employees, but little is told about the radical change that represents moving from a manual procedure to an automated system. The typical case is the data fed into the system: as in a manual scheduling procedure the data are handled by several employees in the company, it is likely that typos and minor errors do not reach to the shop floor, as the human common sense would realize about these errors. In contrast, in a highly automated system, human intervention is minimized and thus the likeliness that errors in the input data translate into wrong production orders increases. The subtle but extremely important change in the role of the scheduler from an executioner to a supervisor must be stressed in the training so most of these errors can be prevented.

3.1.4 Modular design and implementation

While the integration of scheduling with related decisions (such as e.g. lotsizing or material flow control) has attracted a wealth of interest in the last years, such complex models are difficult and expensive from the implementation viewpoint (see e.g. Higgins, 1996). There are a number of reasons for this:

- **Data requirements**. It is obviously that an integrated system would require more data, which is not a trivial issue given the data-intensive nature of the different operations management functions (see e.g. Kathawala and Allen, 1993). In addition, the required quality and granularity of these data may be substantially different, as for instance aggregated estimations of processing times may be more than sufficient for production planning while unacceptable for scheduling. To make an integrated system work, the only solution is to stick to the maximum quality (and quantity) of data required by each of the submodules composing the system, therefore increasing not only the development expense, but also the maintenance costs.

- **Number of stakeholders**. Usually a scheduling decision support system is used by a range of users with different levels of expertise (see e.g. Cowling, 2001). Since more decision makers have to be involved in the process, to agree on constraints and objectives (among other issues) becomes harder. In addition, given the hierarchical nature of some operations management functions (i.e., production planning and production scheduling), it is likely that higher level considerations override the desired functionality, goals and constraints of the scheduling system, which may result in less useful support for the scheduling function.

- **Changing nature of the business**. Given the dynamic nature of the companies, integrated models exhibit a shorter life cycle and may require substantial upgrading when

manufacturing conditions change. A modular design and implementation would increase the reusability of those parts of the system not affected by the changes.

As a conclusion, it seems clear that –despite the potential advantages of a software system virtually capable to handle all operations management -related decisions– simple, modular approaches may be better suited (Pinedo, 2007). The enormous advances in protocols for data exchange among different systems also speak for the convenience of a modular design.

3.1.5 Keep layout modelling simple

Note that layout modelling represents the most persistent abstraction of the scheduling system from the physical factory, as it is likely that this layout would remain stable for longer periods. It is also possibly one of the main issues affecting the validation in the shop floor of the results obtained by the scheduling system, as it would determine the models to be employed, the data required to feed the models, and the solution procedures to be employed for obtaining solutions for the scheduling process. Therefore, these decisions should be carefully regarded.

One advisable hint is to keep the focus on modelling relatively simple layouts –as much as the physical shop floor allows it–. Even if the physical layout of the factory may be very complex, there are a number of reasons indicating the suitability of focusing onto simpler structures, at least on the initial stages of the life cycle of the deployment of the scheduling system. There are at least two reasons for this hint:

- **Dynamic nature of the business**. Whereas theoretical scheduling models assume a set of fixed resources, this is hardly the usual situation in practice. Quite often, additional resources can be moved, purchased, or removed, provoking changes in the physical layout that detailed, low level models cannot easily accommodate (see e.g. Benavides and Carlos Prado, 2002; Freed et al., 2007). Asides, the current shortening of product life cycles and their corresponding technological processes make such changes more likely. This means that detailed scheduling models should be updated, validated, fed with data and put into production at a pace that their maintenance cost could be simply so high that they may never pay off.

- **Preprocessing/What-if analysis**. As mentioned in an earlier section, a preprocessing of the scheduling problem is usually required, which can be used both to set the scope of the scheduling decision problem and for what-if analysis. Pinedo and Yen (1997)'s architecture suggest using a preprocessor for this purpose. It is assumable that this preprocessor may contain a simplified model of the plant, including a simplified layout, as it is used prior to a detailed modeling.

3.2 Data

Here we discuss a number of issues related to data acquisition, manipulation, and maintenance. Scheduling is a data intensive process, and the quality of the data greatly affects the performance of the scheduling models and procedures. Broadly speaking, two types of data are required:

- **Model-based data**. By model-based data we understand all data related to the decision model to be entered into the scheduling system. These typically include the number of machines to be employed at each stage, transportation and other constraints not depending on the specific schedule to be found, etc. These data are relatively stable (at least as compared to the instance-based data) and may be seen as meta-data for the instance-based data to be discussed in the next item.

- **Instance-based data**. These data typically include the processing times of the machines, the routing of the jobs, the cost/income associated to each job, the release and due dates of each job, their weight and/or priority, etc. These data vary from one problem instance to another.

Regarding both types of data, a number of issues are discussed in the following subsections.

3.2.1 Development of a database interface

Usually, some data will be already present in the ERPs or business information systems of the company. However, it is unlikely that all data needed will be present, so data collection becomes an important part in the deployment of a scheduling system. In fact, the data required for a scheduling system may well be dispersed into as much as more than 20 different sources (in this regard see the case study by McKay et al., 1988b).

For introducing and editing model-based data, it may be interesting to develop a graphical interface, but this may be extremely tedious and expensive to maintain for instance-based data. Therefore, time and resources for the development of database interfaces should be allocated in the deployment project. Also note that the construction of such interfaces is usually considered as one of the most time-consuming items for packaged software, and that many maintenance contracts of the business information system vendors and/or IT consultants stipulate that no third-party can access to these data (even if in read-only mode). For such cases, semi-automatic import/export utilities should be the only available option apart from contracting the vendors and consultants for the development of the interface.

Given how hard is to gather the data, an equally detailed system should be placed in order to update and modify existing data. SCADA systems integration is a promising method of automating data maintenance.

Finally, note that advanced models will require extensive datasets that have to be retrieved from the database as needed. Special care should be put in the performance of queries to the databases. Modern distributed tables and threaded queries are necessary.

3.2.2 Data quality

Another source of concern is the quality of the data employed in the scheduling tool. There are case studies in which it is reported that less than 50% of the required data were consistent (see Leachman et al., 1996). Poor data quality is frequently mentioned (see Berglund and Karltun, 2007) as the cause that specific tools and spreadsheets have to be used to complement standard scheduling software tools, and as an explanation of the failure of some scheduling tools' implementations (see Pinedo, 2007). Indeed, one of the tasks of the human scheduler is to filter the data to be employed in the scheduling process (MacCarthy and Wilson, 2001).

3.2.3 Data abstraction

Already mentioned in a previous chapter, a good advice is to abstract internal data structures from the databases and data tables in order to speed up development and implantation. This is heavily related to the data-intensive nature of the scheduling system and may turn useless much of the developments with respect to models and procedures, as the users would distrust such system. Data abstraction serves to focus on the minimum amount of data necessary for the scheduler, so the scheduler's time is not wasted asking for irrelevant data, or deluged with noise in the form of information (McKay and Wiers, 2001).

3.2.4 Keeping interfaces simple

The ERP of the company hardly needs to know the full sequencing with all the start and finish times of the production sequence. Furthermore, providing the ERP with all this data could prove challenging (see Wiers, 2002), as the integration of the ERP with the scheduling system could be extremely costly (in Missbauer et al., 2009 it is described a case in which around 15,000 lines of code were required to support this integration). Probably, only the estimated finishing date for each product is needed. As a result, when connecting a scheduling system with existing ERP software, a good idea is to interface only the most basic information and extend the amount of communication as the needs arise.

3.3 Objectives and Constraints

It is difficult to overestimate the importance of coming to terms with the scheduling objectives. An important part of the of the development of a scheduling system is the discovery of the objectives to be sought. Managers may know what is desirable for the company, but may not be able to formulate operating policies, at least at a scheduling level. Furthermore, the interaction

between operating policies and objectives is often unclear (Bhattacharyya and Koehler, 1998). A probably by-side effect of that is the fact that, at the first glance, virtually all objectives that one may pose could be of interest for the decision maker. It has been often stated that single objective optimisation is controversial in practice, and that several objectives must be taken into account.

Nevertheless, our experience indicates that it is difficult to visualize and to understand more than 2-3 objectives, as the decision maker can hardly make sense of the economic impact of more than 2-3 conflicting operational measures. Therefore, the main issue here is how to prioritize/select among these. There are several approaches to do it, which are discussed in the next subsections.

3.3.1 Prioritisation of objectives

It has been argued that managers may know what is desirable, but may not be able to formulate operating policies for scheduling (see Buxey, 1989; Bhattacharyya and Koehler, 1998). Scheduling encompasses rather short-term decisions, and it is rarely linked to strategic, long-term or medium-term issues. Therefore, considering objectives linked to long-term goals makes little sense for some situations. One typical example would be the maximisation of machine usage. While from a costs accounting perspective machine utilisation helps cutting the unit production costs, this objective could be rarely more critical than fulfilling due dates, which is a pure, short-term objective. Hence, there is no need to consider both objectives simultaneously. Note that we do not claim that machine utilisation (or other medium or long term objectives) are not important: we just stress that operational decisions should be made according to operational measures, and that it is the output of these operational decisions what should feed strategic models, which as a result would determine whether machine utilisation (or any other medium-long term measure) is sufficient to adjust capacity.

In addition, some of the proposed objectives may not be conflicting among them, but are simply different ways to express the goals of the decision makers. Even in some objectives could be theoretically conflicting, it has to be checked whether this happens in the industrial setting where the model is to be deployed. A typical example are number of late jobs and maximum tardiness. It seems clear that one may theoretically reduce the number of late jobs by systematically delaying a small set of jobs, which will in turn generate big tardiness values. However, this alternative cannot be accepted by many companies for which customer reliability is their key competitive advantage. If this results to be the only available option, the company would find a way to renegotiate their commitments and immediately will modify its order acceptance policy to ensure that this situation will not happen again. Therefore, one of the two indicators may suffice as a

scheduling objective. In addition to the additional burden on evaluating a non relevant objective, in a multiobjective optimisation context, it has the additional risk of generating biased solutions, thus not providing the decision maker with an appropriate set of solutions.

3.3.2 Postpone the selection of objectives

Another approach could be to delay the decision on which objectives should be considered. This approach may make sense in some cases, as usually the objectives of scheduling are not an input of the development of the scheduling system, but an output. First of all, it has to be taken into account that, in many cases, the dominant schedule criterion is often scheduling feasibility (Graves, 1981).

Even when the system is finally deployed, the objectives to be considered undergo frequent changes (Cowling, 2003). Therefore, developing the model could serve to better understand the objectives of the scheduling model. In addition, for some scheduling situations, finding a feasible (and understandable) solution may suffice (see e.g. Cowling, 2003; Gao and Tang, 2008), at least during the first stages of the development, therefore the decision on the objectives may not be so critical at the early stages. While developing the scheduling model, the implicit insights of the decision maker become explicit and he/she can state the objectives in a clearer way.

3.3.3 Objectives versus constraints

In practice, some objectives can be safely transferred into constraints, as it has been observed that in the real world schedulers override capacity constraints and therefore the scheduling tool should allow the violation of these constraints (McKay and Wiers, 2001). In addition, our experience is that this may apply to many penalty or cost (profit) -related objectives, as it may not be so easy to derive an accurate penalty/cost (profit) function. A typical example are due dates: some companies accept the violation of committed due dates, and recognise that this should imply a sort of penalty which is approximately related to the number of violations (service level) and to the difference between the promised due dates and the completion times (average waiting time). However, establishing such relation in a meaningful function that adequately weights them and can be easily calculated for all jobs and customers is not so easy. Therefore, a simple alternative is to establish a maximum tardiness allowed and/or a maximum fraction of jobs late. Since most production departments include objectives or indicators linked to service level and lead times, obtaining these maximum levels of allowance is straightforward in many situations.

3.3.4 "New" types of constraints

The determination and satisfaction of a large variety of constraints is considered to be the crux of scheduling (Fox and Smith, 1984). The number of constraints that can be identified in the shop floor can be extremely high (see e.g. the survey on scheduling practices for 40 schedulers by McKay et al., 1988a where more than 200 constraints are identified). Obviously, not all of them are relevant for scheduling. Indeed, schedulers spend 80 to 90% of their time determining the constraints of the environment that will affect their scheduling decisions (Crawford and Wiers, 2001; Grant, 1986).

While the scheduling literature is rich in dealing with certain types of constraints, there are constraints that are so common in real environments that it is surprising that most existing models do not deal with them. In the following we mentioned some of them grouped in different categories:

- *Machine constraints.* Apart from specific machine constraints, most shops are characterised by the non availability of machines for all purposes. On one hand, not all machines may be amenable to process all jobs (or even if this is the case from a technological viewpoint, there may be economic reasons preventing that). Therefore, there is a problem of machine eligibility. On the other hand, in most cases, eligible machines are not available during all planning period. This is not only motivated by breakdowns or planned maintenance, but also because machines are busy processing already scheduled jobs. However, existing research on this topic is almost inexistent.

- *Staff constraints.* Staff is almost an ignored resource in many scheduling models (see in this regard the results of the survey by Laforge and Craighead, 2000). However, it is of utmost importance for many real shops. Even in automatic or semi-automatic processes, staff plays an important role in setup times. Existing literature on scheduling with additional resources is rather scarce, there are some studies for parallel machines like the one by Chen (2006) but not for complex scheduling problems.

- *Routing constraints.* Many shop floors require some type of job reentrance and/or precedence constraints. The reasons vary from pure process-related to quality (including scraps and the need for reworking). The complexity of this type of resources (usually affecting a small percentage of the jobs in the system) should be balanced against the effort of developing a system that explicitly takes them into account.

- *Transportation constraints.* Transportation times in between stages and the control of the AGVs is also necessary for many systems.

- *Capacity constraints.* Storage capacity (number of boxes) in between each stage is also a key concern as in some cases this is a limited resource.

Note that we do not claim that the previous constraints have not been studied at all. However, it can be safely said that most studies involve simplistic production layouts (often single machine problems) and in any case, the previous set of constraints has been seldom considered in a simultaneous way, even less so for complex scheduling settings.

3.4 Solution procedures

A great effort in the scheduling field has been devoted to solution procedures, at least from the most theoretical side. It is dubious that this interest is similar in practice, as there are case studies indicating that "mathematical algorithms were not considered a priority" (McKay and Black, 2007). In this line, Portougal and Robb (2000) find that in a survey of 9 manufacturing firms, in only one of them a complex scheduling algorithm would help.

A general problem is that most existing solution procedures have been tightly linked to the models, therefore generally resulting in algorithms with low performance outside the original models for which they were conceived, if applicable at all. Since no algorithm can outperform the rest for all scheduling models (no free lunch theorem), building specific algorithms may be justified from a theoretical viewpoint, but it should be also clear that no scheduling system can 1) store the myriad of specific algorithms, and 2) select the most suitable one among them for any scheduling decision. In addition, we have already discussed the relatively short life-cycle of scheduling models in view of the extremely dynamic nature of manufacturing. As a consequence, the advantages of designing specific algorithms in terms of quality of the solutions should be balanced against their cost of development and deployment. In addition, we note the following issues:

3.4.1 Focus on feasibility

It is quite necessary to balance the effort in developing the solution procedures against their advantages. As mentioned before, in some cases the problem is so restricted that there are few feasible solutions (Graves, 1981; Benavides and Carlos Prado, 2002; Cowling, 2003). In these cases, most of the computational effort may be wasted in checking that no better feasible solutions are available. In addition, in some cases, the objective function is rather flat as compared to "classical" objective functions. Again, in these cases, most of the computational effort is spent on confirming optimal values. This speaks for the need of balancing the effort in developing the solution procedures, particularly taking into account the aforementioned changing nature of the objectives sought.

3.4.2 Take into account the available decision time

The algorithms employed have to make the best use of the available running (decision) time (Cowling, 2003), so there is need of developing solution procedures whose quality of solutions is scaled with the decision interval. As the decision interval is very context specific (it may not only vary from company to company, but also within a single company, depending on the shift, workload, etc.), it would be extremely interesting to build algorithms with a performance that is (roughly) linear with time (i.e., they do not stall after a number of iterations or require very long decision intervals to obtain acceptable solutions).

Note that the focus towards this type of algorithms also implies the need of re-assessing the way some computational experiences of scheduling algorithms are carried out: Usually, a new solution procedure for a specific scheduling problem is labeled as "efficient" because it yields better solutions than existing ones when all of them are allowed the same CPU time. However, the relative performance of this new solution procedure for different CPU times is unknown.

3.4.3 Providing a set of alternative solutions

Even in the best defined industrial scheduling problems, some aspects cannot be captured by the objectives or the constraints of the problem, even if they remain important in decision maker's eyes. Therefore, it is interesting providing him/her with a set of "good" alternative solutions. This can be done by using the support of different algorithms for the problem (see e.g. Tang and Wang, 2008), for which it is clear that the algorithms should be fast enough to fit within the decision interval.

An alternative option is to provide the scheduler with at least all solutions with equal objective values found by the algorithm. This would clearly help him/her to safely pick the one fitting best into these "implicit goals", and at the same time, it would allow him/her to make these goals more explicit so they can later be incorporated into the system.

3.4.4 Transparency of solution procedures

A special effort has to be made to develop a transparent system who can be understood by the users (see Wiers, 1997; Baek et al., 1999; Wiers, 2001; Freed et al., 2007). It does not seem reasonable to believe that decision makers with years of practice in an specific production shop floor would immediately accept any solution packaged into a scheduling system (a "black-box" sequence, see Cowling, 2003), particularly if it does not follows his/her intuition. Also note that the need for transparency increases where the scheduling tasks is perceived as critical (Wiers, 2001).

Even if this issue could be fixed (at least in part) by a better training of the schedulers

(Laforge and Craighead, 2000), the understanding of the logic behind the solution procedures, even at a rough level, will surely increase trust in the system and will incentivate its use.

As a consequence, solution procedures based on well-defined manufacturing principles may provide a more transparent system. Some of these principles could be the following:

- Focusing on bottleneck resources. Focusing on the bottleneck is a well-understood principle that helps managers to overcome the complexity of the factory and concentrate of the part of the shop floor that greatly impacts the performance of the system.

- Aggregation of non critical resources. Many heuristic procedures are based on the principle of aggregating machines and/or jobs in order to apply a solution procedure that works well –or it is even optimal– for the resulting (aggregated) problem. When these machines/jobs are not critical, this principle is easy to be understood.

- Employing simple dispatching rules. For some highly constrained shops, there is little space for improvement, and simple dispatching rules may provide good solutions. Even if this is not the case, dispatching rules are well-known (and employed) by schedulers, so they can be used as a benchmark to assess the improvement of more sophisticated algorithms and to help schedulers in deciding whether this increasing complexity (and thus a loss of transparency) is worth the effort.

- Implement heuristics extracted from schedulers' experience. In researching what schedulers did (see MacCarthy and Wilson, 2001), a number of qualitative heuristics and "rules of thumb" have been encountered, such as scheduling certain type of job as the next job to be processed in a machine that has just been repaired. While there are several studies to assess whether these "heuristics" are an asset or a liability (see e.g. Steffen, 1986; Kanet and Adelsberger, 1987; McKay et al., 2000), a good strategy may be to implement in the system some of these procedures. This would not only provide the scheduler with a link between his/her reasoning and the system, but also would eventually help him/her to evaluate the rest of the algorithms provided by the system.

- Restricting moves to non critical jobs. This is another classical strategy, as it may help in marginal improvements while keeping the feasibility (or desired properties) of the initial solution.

In addition, such procedures generally require less data and are more robust to shop floor variability than their more complex counterparts. The widespread success of dispatching rules despite their relatively poor performance speaks for the need of carefully balancing the sophistication in the design of solution procedures against their understanding and acceptance by decision makers. In this sense, it is perhaps interesting to draw the attention onto research on general frameworks to develop heuristics, such as the one by Pinedo and Yen (1997).

3.4.5 Design of algorithms

Many approximate algorithms (i.e., metaheuristics) employ a number of parameters that must be tuned in order to accomplish a good performance. The usual procedure for tuning the algorithms in a laboratory is to collect a set of instances and try different values of their parameters (using a Design of Experiments) and pick the one yielding the best performance. While these procedures may not represent a big issue in a laboratory, it is somehow problematic in a real setting. First, such set of instances may not be available, or it may be outdated with respect e.g. to the processing times of the jobs to be scheduled in the future. Secondly, the logic of these parameters may not be clear to the decision maker and thus may be use them, or misuse them. Therefore, we believe that either algorithms with many parameters should be discarded, or the details or their tuning should be made transparent to the decision maker (see e.g. Wiers, 2001). Clearly, this does not include the running time of the algorithm, as we have already discussed its importance.

3.4.6 Manipulation of solutions

If we accept that no mathematical model can pretend capturing all possible scenarios and exceptions, it follows that we should allow the decision maker to manipulate the solutions obtained by the different procedures, as there may be extraneous constraints and/or objectives which are not considered by the system and must be satisfied by the resulting solution (Bensana et al., 1986). In addition, surveys indicate that there are also external reasons requiring the frequent override or adjusting of the schedules, such as later or incorrect deliveries or customer order changes (Halsall et al., 1994).

While the scheduling literature is practically unanimous in pointing at this functionality (see e.g. Collinot et al., 1988; Bitran and Tirupati, 1988; Numao and Morishita, 1989; Prietula et al., 1994; Sauer and Bruns, 1997; T'kindt et al., 2005; Pinedo, 2007), several approaches could be adopted to do so. One option would be to allow drag and drop the solutions on the Gantt chart. Another option, more complex, would be to allow partly to freeze jobs and/or partial schedules. This latter option is also very suitable for considering jobs that are being manufactured, and for reducing the nervousness of changes in the schedule, and it brings the interesting problem of the advantages of rescheduling / partly scheduling the non frozen jobs. Unfortunately, this problem has not been –to the best of our knowledge– addressed in the scheduling literature, but lessons from the application of production planning systems, such as MRP and MRPII could be learnt.

As mentioned in Bitran and Tirupati (1988), the excessive manipulation of the solutions proposed by the system may result into worsening the original objective function. As a mean to control and to balance this issue, in Framinan and Ruiz (2010) it was proposed that the tool provides reports indicating the deviation of the so-modified schedules with respect to the ones suggested by the algorithms.

4 Conclusions

In this paper, we have focused on an area that has received relatively little attention in the scheduling field, at least as compared to the wealth of literature dealing with scheduling models and procedures. However, in order to put these models and procedures into practice, one must adequately capture the details of a company's scheduling process, which usually entails modeling scenarios, constraints and objectives not often treated in the literature. In addition, the resulting models and solution procedures should be embedded into a scheduling system that feeds them with reliable and timely data and provides an efficient way to interact with the users. This gives rise to the following interrelated questions: Which are the components of a scheduling system?, and, How a scheduling system can be developed and deployed? Regarding this second question, we propose in this paper a number of guidelines to help driving the design and implementation of scheduling systems based on the previous analysis of the literature on case studies and on our own experience.

As mentioned in the beginning of the paper, we do not claim that our guidelines are the only way (or even the best way) to address the aforementioned issues, although they have been validated both through literature analysis and through industrial practice. By presenting them, we hope to enrich the rather scarce literature on the topic and thus help bridging the gap between the development of scheduling models and procedures, and their implementation in a real-life industrial settings. This area opens a number of extremely important issues for future research, most of which have been already mentioned throughout the paper.

Finally, note that we will not make any distinction among customised and packaged scheduling systems, although several authors state that current packaged software is unable to adequately support the needs of production scheduling (Dumond, 2005). Regardless this controversy, most of the guidelines apply to both types of systems, and just a few of them (particularly those related to the design of solution procedures) may, in principle, be more useful for customised systems.

References

Baek, D., Oh, S., and Yoon, W. (1999). A visualized human-computer interactive approach to job shop scheduling. *International Journal of Computer Integrated Manufacturing*, 12(1):75–83.

Benavides, J. and Carlos Prado, J. (2002). Creating an expert system for detailed scheduling. *International Journal of Operations and Production Management*, 22(7-8):806–819.

Bensana, E., Correge, M., Bel, G., and Dubois, D. (1986). An expert-system approach to industrial job-shop scheduling. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1645–1650.

Berglund, M. and Karltun, J. (2007). Human, technological and organizational aspects influencing

the production scheduling process. *International Journal of Production Economics*, 110(1-2):160–174.

Bhattacharyya, S. and Koehler, G. (1998). Learning by objectives for adaptive shop-floor scheduling. *Decision Sciences*, 29(2):347–375.

Bitran, G. R. and Tirupati, D. (1988). Development and implementation of a scheduling system for a wafer fabrication facility. *Operations Research*, 36(3):377–.

Buxey, G. (1989). Production scheduling: Practice and theory. *European Journal of Operational Research*, 39(1):17–31.

Chen, J.-F. (2006). Unrelated parallel machine scheduling with secondary resource constraints. *International Journal of Advanced Manufacturing Technology*, 26(3):285–292.

Collinot, A., Le Pape, C., and Pinoteau, G. (1988). SONIA: A knowledge-based scheduling system. *Artificial Intelligence in Engineering*, 3(2):86–94.

Cowling, P. (2001). Design and implementation of an effective decision support system: A case study in steel hot rolling mill scheduling. In MacCarthy, B. and Wilson, J., editors, *Human performance in planning and scheduling*, pages 217–230. Taylor & Francis.

Cowling, P. (2003). A flexible decision support system for steel hot rolling mill scheduling. *Computers and Industrial Engineering*, 45:307–321.

Crawford, S. and Wiers, V. (2001). From anecdotes to theory: reviewing the knowledge of the human factors in planning and scheduling. In MacCarthy, B. and Wilson, J., editors, *Human performance in planning and scheduling*, pages 15–44. Taylor & Francis.

Dudek, R. A., Panwalkar, S. S., and Smith, M. L. (1992). The lessons of flowshop scheduling research. *Operations Research*, 40(1):7–13.

Dumond, E. (2005). Understanding and using the capabilities of finite scheduling. *Industrial Management & Data Systems*, 105(4):506–526.

Ford, F. N., Bradbard, D. A., Ledbetter, W. N., and Cox, J. F. (1987). Use of operations research in production management. *Production and Inventory Management*, 28(3):59–62.

Fox, M. S. and Smith, S. (1984). ISIS: A knowledge-based system for factory scheduling. *Expert Systems Journal*, 1(1).

Framinan, J. M. and Ruiz, R. (2010). Architecture of manufacturing scheduling systems: Literature review and an integrated proposal. *European Journal of Operational Research*, 205(2):237–246.

Freed, T., Doerr, K., and Chang, T. (2007). In-house development of scheduling decision support systems: Case study for scheduling semiconductor device test operations. *International Journal of Production Research*, 45(21):5075–5093.

Gao, C. and Tang, L. (2008). A decision support system for color-coating line in steel industry. In *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2008*, pages 1463–1468.

Grant, T. J. (1986). Lessons for O.R. from A.I.: A scheduling case study. *Journal of the Operational Research Society*, 37 (1):41–57.

Graves, S. C. (1981). A review of production scheduling. *Operations Research*, 29(4):646–675.

Halsall, D. N., Muhlemann, A. P., and Price, D. H. (1994). A review of production planning and scheduling in smaller manufacturing companies in the uk. *Production Planning & Control*,

5(5):485–.

Higgins, P. G. (1996). Interaction in hybrid intelligent scheduling. *International Journal of Human Factors in Manufacturing*, 6(3):185–203.

Kanet, J. J. and Adelsberger, H. H. (1987). Expert systems in production scheduling. *European Journal of Operational Research*, 29(1):51–59.

Kathawala, Y. and Allen, W. (1993). Expert systems and job shop scheduling. *International Journal of Operations & Production Management*, 13(2):23–35.

Kerr, R. M. (1992). Expert systems in production scheduling: Lessons from a failed implementation. *Journal of Systems and Software*, 19(2):123–130.

Knolmayer, G., Mertens, P., and Zeier, A. (2002). *Supply chain management based on SAP systems*. Springer.

Laforge, R. and Craighead, C. (2000). Computer-based scheduling in manufacturing firms: Some indicators of successful practice. *Production & Inventory Management Journal*, 41(1):29–34.

Leachman, R., Benson, R., Liu, C., and Raar, D. (1996). Impress: An automated production-planning and delivery-quotation system at harris corporation - semiconductor sector. *Interfaces*, 26(1):6–37.

Ledbetter, W. N. and Cox, J. F. (1977). Operations research in production management: An investigation of past and present utilization. *Production and Inventory Management*, 18(3):84–91.

MacCarthy, B. and Wilson, J. (2001). *Human performance in Planning and Scheduling*. Taylor & Francis.

MacCarthy, B. L. and Liu, J. (1993). Addressing the gap in scheduling research: A review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79.

McKay, K., Morton, T., Ramnath, P., and Wang, J. (2000). 'aversion dynamics' scheduling when the system changes. *Journal of Scheduling*, 3(2):71–88.

McKay, K., Safayeni, F., and Buzacott, J. (1988a). Job-shop scheduling theory: What is relevant? *Interfaces*, 18(4):84–90.

McKay, K. and Wiers, V. (2001). Decision support for production scheduling tasks in shops with much uncertainty and little autonomous flexibility. In MacCarthy, B. and Wilson, J., editors, *Human Performance in Planning and Scheduling*, pages 165–178. Taylor & Francis.

McKay, K. N. and Black, G. W. (2007). The evolution of a production planning system: A 10-year case study. *Computers in Industry*, 58(8-9):756–771.

McKay, K. N., Pinedo, M. L., and Webster, S. (2002). Practice-focused research issues for scheduling systems. *Production and Operations Management*, 11(2):249–258.

McKay, K. N., Safayeni, F. R., and Buzacott, J. A. (1988b). Job-shop scheduling theory: What is relevant? *Interfaces*, 4(18):84–90.

Meador, C. L. and Ness, D. N. (1974). Decision support systems - application to corporate planning. *Sloan Management Review*, 15(2):51–68.

Missbauer, H., Hauber, W., and Stadler, W. (2009). A scheduling system for the steelmaking-continuous casting process. a case study from the steel-making industry. *International Journal of Production Research*, 47(15):4147–4172.

Numao, M. and Morishita, S. (1989). A scheduling environment for steel-making processes. *Proceedings of the 5th Conference on Artificial Intelligence Applications*, pages 279–286.

Olhager, J. and Rapp, B. (1995). Operations research techniques in manufacturing planning and control systems. *International Transactions in Operational Research*, 2(1):29–43.

Pinedo, M. L. (2007). *Planning and Scheduling in Manufacturing and Services*. Springer, Berlin, Third edition.

Pinedo, M. L. and Yen, B. P.-C. (1997). On the design and development of object-oriented scheduling systems. *Annals of Operations Research*, 70(1):359–378.

Portougal, V. and Robb, D. (2000). Production scheduling theory: Just where is it applicable? *Interfaces*, 30(6):64–76.

Prietula, M., Hsu, W., Ow, P., and G.L., T. (1994). Macmerl: Mixed-initiative scheduling with coincident problem spaces. In Zweben, M. and Fox, M., editors, *Intelligent Scheduling*, pages 655–682. Morgan Kaufmann.

Reisman, A., Kumar, A., and Motwani, J. (1997). Flowshop scheduling/sequencing research: A statistical review of the literature, 1952-1994. *IEEE Transactions on Engineering Management*, 44(3):316–329.

Sauer, J. and Bruns, R. (1997). Knowledge-based scheduling systems in industry and medicine. *IEEE Expert*, Janury-February:24–31.

Steffen, M. S. (1986). A survey of artificial intelligence-based scheduling systems. In *Proceedings of the Fall Industrial Engineering Conference*.

Storer, R., Wu, S., and Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38:1495–1509.

Tang, L. and Wang, G. (2008). Decision support system for the batching problems of steelmaking and continuous-casting production. *Omega*, 36(6):976–991.

T'kindt, V., Billaut, J.-C., Bouquard, J.-L., Lenté, C., Martineau, P., Néron, E., Proust, C., and Tacquard, C. (2005). The e-OCEA project: towards an internet decision system for scheduling problems. *Decision Support Systems*, 40(2):329–337.

Vernon, C. (2001). Lingering amongst the lingerie: An observation-based study into support for scheduling at a garment manufacturer. In MacCarthy, B. and Wilson, J., editors, *Human performance in planning and scheduling*, pages 135–163. Taylor & Francis.

Webster, S. (2001). A case study of scheduling practice at a machine tool manufacturer. In MacCarthy, B. and Wilson, J., editors, *Human performance in planning and scheduling*, pages 67–81. Taylor & Francis.

Wiers, V. (1997). Human-computer interaction in production scheduling: Analysis and design of decision support systems for production scheduling tasks.

Wiers, V. (2001). Design of knowledge-based scheduling system for a sheet material manufacturer. In MacCarthy, B. and Wilson, J., editors, *Human Performance in Planning and Scheduling*, pages 201–215. Taylor & Francis.

Wiers, V. (2002). A case study on the integration of APS and ERP in a steel processing plant. *Production Planning & Control*, 13(6):552–560.

Wiers, V. and Van Der Schaaf, T. (1997). A framework for decision support in production scheduling tasks. *Production Planning & Control*, 8(6):533–544.