3rd International Conference on Industrial Engineering and Industrial Management
XIII Congreso de Ingeniería de Organización
Barcelona-Terrassa, September 2nd-4th 2009

Guidelines for deploying and implementing scheduling systems*

Jose M Framiñán[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

Keywords: Deployment, Implementation, Scheduling systems.

1. Introduction

While the literature on scheduling models and solution procedures is extensive, very little has been written on how to bring these models and procedures into practice. This has given rise to the so-called "gap" between the theory and practice of scheduling (see McCarthy and Liu 1993), which has been widely documented in several studies, such as e.g. Ford et al. (1987). In order to close this gap between scheduling models and procedures, and their implementation in a real setting, the former should be translated into a piece of software supporting scheduling decisions in a company. This implies carrying out a software development process to obtain a final product, i.e. a scheduling system at work. As such a 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.

Despite the importance of the process of developing such systems, scheduling research has often overlooked the topic, as there are hardly references providing guidance or recommendations to successfully accomplish the development of a manufacturing scheduling system, or at least case studies describing this development process for a particular application so lessons and insights for future developments can be learned.

Our paper is aimed towards these two important issues. To do so, we first review the existing literature on case studies regarding the application of scheduling systems. From this review, we derive and classify a number of guidelines for developing scheduling models for industrial practice based on our experience and on the analysis of the relevant literature. While we do not claim that the proposed guidelines are universally valid nor should be strictly followed, we hope that they will help orienting the design of scheduling models towards a greater applicability.

2. Background

Scheduling systems can be considered as a particular case of business information systems. Usually, business information systems can be divided into packaged (standard) software, and customised software (see e.g. Kirchmer 1999). Although there are several available standard scheduling systems, the technological peculiarities of different production environments make

* José M. Framiñán is partially funded by the Spanish Ministry of Science and Innovation, under project "Advanced Systems for Integrated Order Management" with reference DPI2007-61345 /DPI. Rubén Ruiz is partially funded by the Spanish Ministry of Science and Innovation, under the project "SMPA - Advanced Parallel Multiobjective Sequencing: Practical and Theorerical Advances" with reference DPI2008-03511/DPI.

difficult to come up with a general purpose scheduling approach (Brandimarte 2000), and quite often the code developed for the customisation of a packaged scheduling software turns to be more than half the code of the final version (Pinedo 2005). As a result, the development of a generic scheduling tool that can be widely installed has eluded the many vendors who have tried (McKay, 2000). Therefore, in the following we will focus on customised scheduling systems, although most of the discussion could also apply to standard scheduling systems.

Broadly speaking, the development of a customised information system encompasses the following activities, which are independent from the adopted software development process (Kurbel, 2008):

- Requirement analysis (or requirements engineering), i.e. determining the needs of the information system to be developed. Requirements are usually classified into functional requirements (those defining a function of the software), and non functional requirements (those imposing constraints on the system such as performance requirements, security, or reliability).
- System design, i.e. providing a conceptual solution to the requirements that have been identified. Usually, system design is broken down into the description of the software components of the system and their relationships (what it is called the *architecture* of the system), and the detailed design of these software elements (Jacobson 1999).
- System implementation, i.e. transforming the conceptual solution into a piece of software.
- System testing, i.e. carrying out a validation and verification of the implemented system.

Although customised manufacturing scheduling systems are, by definition, different for each company, it is clear that some activities in the development process may be common to all companies, as they refer to high-level descriptions of the purpose of the system. This is illustrated in Figure 1, where generic (common) and specific activities are depicted. While there are a number of company-dependent purposes for which a scheduling system can be implemented (see e.g. Aytug 2005 for a classification of the different purposes of scheduling), most scheduling systems share a number of requirements, as all of them must have a number of common functionalities. Since these requirements are reflected into components of the architecture of the system, it is clear that some parts of this architecture may also be common to most manufacturing scheduling systems. The re-use of an efficient, validated architecture instead of designing a new system from scratch has a number of advantages:

- It shortens the development cycle, as it saves time (and money) that otherwise should have been allocated to requirements analysis and design.
- It ensures that the main functionalities of a scheduling system are adequately covered, thus acting the architecture both as checklist and design guide for the developers.
- It allows the re-utilization for future systems of part of the code developed, provided that the architecture is described in terms of blocks or function-specific modules.

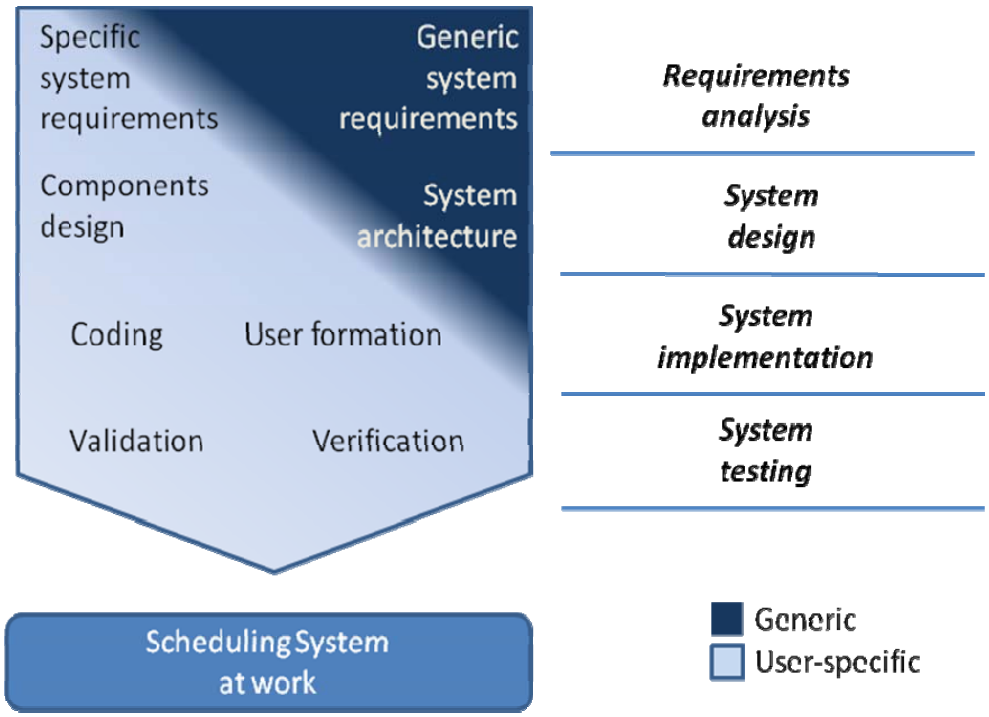

Figure 1. Generic and specific activities in the development of a manufacturing scheduling system.

In this paper, we are interested in the so-called generic activities, from which we can obtain a number of insights regarding the deployment of manufacturing scheduling systems.

There are several papers dealing with the description of manufacturing scheduling systems and their implementation. These contributions range from high-level descriptions of the main components of a general-purpose architecture of these systems (such as in e.g., Pinedo and Yen 1997, Ecker 1997, T'kindt et al. 2005, or Yen 2004), to detailed discussions about specific systems for a particular context (such as e.g., Numao and Morishita 1989, or Hadavi et al. 1990). Some papers concentrate on describing the structure of data handled by different scheduling systems with certain level of detail (such as in Blazewicz et al. 2001 or Rossi et al. 1998), while others describe a hierarchy of classes referring to some of the persistent elements mentioned before (such as Sauer 1993, Maturana et al. 1997, or Pinedo 2007). Given the lack of homogeneity among these contributions, a paper-by-paper description would offer little insight on the components discussed by the different authors. Instead, we have analysed these contributions and derived a number of guidelines (which are discussed in the next section), citing the corresponding references whenever appropriate. These guidelines also serve as conclusions of the analysis.

3. Guidelines

In this section we provide a number of conclusions derived from the analysis of the related literature, and from our hands-on experience. We have grouped them into specific topics. These are:

– **Model layout**. The focus should be kept on modelling relatively simple layouts. 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 scheduling system. These reasons include:

- *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. Therefore, even if the final goal would be the detailed modelling of the plant layout, a simpler model would suffice for launching the scheduling system and making the first results of its adoption visible. Despite their potential shortcomings, the literature on implementation of business information systems 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 most scheduling systems.
- *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. 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 in strict terms. However, most of these settings result in much simpler layouts for most variants after a Pareto (A-B-C) analysis. As a result, it seems sensible to develop a system to schedule "just" 80% of the jobs rather than considering a much more complex layout in order to handle the other remaining 20%, particularly if we adopt the aforementioned incremental approach.
- *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. 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's architecture (Pinedo and Yen 1997) suggests using a preprocessor for this purpose. It is assumable that this preprocessor may contain a simplify model of the plant, including a simplified layout, as it is used prior to a detailed modelling. The need of the what-if analysis is also mentioned by McKay and Wiers (2003).

– **Data acquisition and manipulation**. Many authors (including most of the references cited in the previous section) have consistently stated how hard is to collect and keep all data needed by a complex scheduling system. The following issues must be addressed:

- *Development of a database interface*. Usually, some data will be already present in the ERPs of the company. However, it is unlikely that all data needed are present. As a result, both an interface with the database as well as a graphical interface for new data introduction is necessary. A good advice is to abstract internal data structures from the databases and data tables in order to speed up development and implementation.
- *Maintenance of data*. 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. This is

particularly true if the scope of the scheduling system includes the shop floor control of the schedules (see e.g. McKay and Wiers 2003).

- *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. 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. An analysis of the suitable interfaces for manufacturing scheduling systems is carried out by Higgins (1996).
- *Performance*. 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.

– **Objectives**. It is difficult to overestimate the importance of coming to terms with the scheduling objectives. At the first glance, virtually all objectives that one may pose could be of interest for the decision maker, as single objective optimisation is controversial in practice. Nevertheless, our experience indicates that it is difficult to visualize and 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:

  - *Prioritisation of objectives*. 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.
  - *Taking out non-conflicting objectives*. 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. Typical examples are the 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 multi-objective optimisation context, it has the additional risk of generating biased solutions, thus not providing the decision maker with an appropriate set of solutions.

- *Postponement*. 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. 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, 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.
- *Transforming objectives into constraints*. In practice, some objectives can be safely transferred into constraints. 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.

– **Solution procedures**. A great effort in the scheduling field has been devoted to solution procedures, at least from the most theoretical side. 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 outperforms the rest for all scheduling models, 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 algorithm in terms of quality of the solutions should be balanced against their cost of development and deployment In addition, we note the following issues:

  - *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 (Hopp and Spearman 1996). 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 this case most of the computational effort is spent on confirming optimal values. This not only speaks for the need of balancing the effort in developing the solution procedures, but also for the need of developing solution procedures whose quality of solutions is

scaled with the decision interval. As the decision interval is very context specific (also within a single company, depending on the shift, workload, etc.), it would be extremely interesting to build algorithms that their performance 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). 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 labelled as ``efficient'' problem 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.

- *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'' solutions, or 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.
- *Understanding solution procedures*. 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, particularly if it does not follows his/her intuition. Therefore, the understanding of the logic behind the solution procedures, even at a rough level, will surely increase trust in the system and will incentivize its use. As a consequence, solution procedures based on well-defined manufacturing principles (such as focusing on bottleneck resources, aggregation of non critical resources, restricting moves to non critical jobs, etc.) are the key for acceptance. 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).
- *Design of algorithms*. Many approximate algorithms (i.e., meta-heuristics) 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 (see bullet before). 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. Clearly, this does not include the running time of the algorithm, as we have already discussed its importance.

- *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. This aspect is mentioned, among others, by Pinedo and Yen (1997).Several approaches could be adopted. One option would be to allow drag and drop the solutions on the Gantt chart. Another option, more complex, would be to allow partly freezing 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. In this regard, lessons from the application of production planning systems, such as MRP and MRPII could be learnt.
- *Using already available technology*. As more and more complex scheduling settings result in slow algorithms, possible solutions are parallelisation and cooperative computing. By using grids in a much simpler level, the available multiple CPU cores of modern computers, results of very high quality can be obtained in a fraction of CPU time. Similarly, very complex scheduling problems can be solved in fast CPU times. Research in this field is rather scarce as compared on the large body of research on ``sequential'' scheduling.

– **Modular design**. While the integration of scheduling with related decisions (such as e.g.\ lot sizing or material flow control) has attracted a wealth of interest in the last years, such complex models are difficult from the implementation viewpoint. First, they require more data. Also, more decision makers could be involved in the process, making more difficult to agree on constraints and objectives. Finally, given the dynamic nature of companies, integrated models exhibit a shorter life and require substantial upgrading when manufacturing conditions change. Instead, simple, modular approaches may be reusable.

– **Constraints**. While 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.
- *Staff constraints*. Staff is almost an ignored resource in many scheduling models. However, it is of utmost importance for many real shops. Even in automatic or semi-automatic processes, staff plays an important role in set ups times. Existing literature on scheduling with additional resources is rather scarce. More specifically, there are some studies for parallel machines like the one by Chen (2006) but not for complex scheduling problems like the ones approached in this paper.
- *Routing constraints*. Many shop floors require some type of job re-entrance and/or precedence constraints. The reasons vary from pure process-related to quality (including scraps and the need for reworking).

- *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.

References

Aytug, H. and Bhattacharyya, S. and Koehler, G. J. and Snowdon, J. L. (1994). A review of Machine Learning in Scheduling, IIE Transactions on Engineering Management 41 (2), 165-171.

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., and Weglarz, J. (2001). Scheduling Computer and Manufacturing Processes, Springer.

Brandimarte, P. and Rigodanza, M. and Roero, L. (2000). Conceptual modeling of an object-oriented scheduling architecture based on the shifting bottleneck procedure, IIE Transactions, 32 (10), 921-929.

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

Ecker, K. and Gupta, J.N.D. and Schmidt, G. (1997). A framework for decision support systems for scheduling problems, European Journal of Operational Research, 101 (3), 452-462.

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, 59-62.

Hadavi, K., Shahraray, M.S. and Voigt, K. (1990). ReDS-A dynamic planning, scheduling, and control system for manufacturing, Journal of Manufacturing Systems, 9 (4), 332-344.

Higgins, P. G. (1996). Interaction in Hybrid Intelligent Scheduling, International Journal of Human Factors in Manufacturing, 6 (3), 185-203.

Hopp, W. J. and Spearman, M. L. (1996). Factory physics. Foundations of manufacturing management, Irwin.

Jacobson, I. and Booch, G. and Rumbaugh, J. (1999). The Unified Software Development Process, Addison-Wesley Professional.

Kirchmer, M. (1999). Business Process Oriented Implementation of Standard Software, Springer.

Kurbel, K.E. (2008). The making of Information Systems, Springer.

Maturana, F., Gu, P., Naumann, A. and Norrie, D. H. (1997). Object-oriented job-shop scheduling using genetic algorithms, Computers in Industry, 32 (3), 281-294.

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, 59-79.

McKay, K.N. and Buzacott, J.A. (2000). Application of computerized production control systems in job shop environments, Computers in Industry, 42 (2), 79-97.

McKay, K.N. and Wiers, V.C.S. (2003). Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory, Computers in Industry, 50 (1), 5-14.

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

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.

Pinedo, M.L. (2007). Planning and Scheduling in Manufacturing and Services, Springer.

Rossi, P., Diaz, C., Fruttero, P., Vittori, C., Rico, M. and Chiotti, O. (1998) Decision support systems generator for industrial companies. Module III: Scheduling support system, Computers \& Industrial Engineering, 35 (1-2), 311-314.

Sauer, J. (1993). Meta-scheduling using dynamic scheduling knowledge, in: Scheduling of Production Processes, Edited by Dorn, J. and Froeschl, K., 151-162, Ellis Horwood, Upper Saddle River, N.J.

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.

Yen, B.P.-C. and Wu, O.Q. (2004). Internet scheduling environment with market-driven agents, IEEE Transactions on Systems, Man and Cybernetics, Part A, 34 (2), 281-289.