Nine Interaction Guidelines for the Design of Human Computer Interaction in Detailed Scheduling Systems

Anna Prenzel, Georg Ringwelski

Hochschule Zittau/Görlitz
Fakultät Elektrotechnik und Informatik
Brückenstraße 1
02826 Görlitz
aprenzel@hszg.de
gringwelski@hszg.de

Abstract: With the help of modern software, the scheduling process in many fields can be automated nearly completely. However, these systems often provide insufficient interaction possibilities for the scheduler to incorporate his own expert knowledge. Finding and implementing suitable planning decisions is often a time-consuming and error-prone process that is not supported by the computer. This deficiency can impair the practicability of the schedules and hence the benefit of the planning system considerably. In this paper, we deduce 9 interaction guidelines that form a concept for the design of human-computer-interaction in scheduling systems. Unlike approaches that extend general HCI guidelines we derive the rules from recent research on human decision making. From that perspective we define computer support for both the formation and the implementation of planning decisions by the scheduler. The results of a user test show that the concept allows an effective incorporation of the human scheduler. Furthermore the concept can be applied to any detailed scheduling problem.

1 Introduction

1.1 Application area

The interaction guidelines described in this paper can be applied in any detailed scheduling system that supports the assignment of *jobs* to *resources*. Examples are the assignment of shifts to staff members, production items to machines or transport orders to vehicles. Detailed scheduling systems serve to make operative planning decisions (rather than tactical or strategic planning decisions, see [SFG$^+$12]) prescribing the temporal arrangement of jobs within the schedules of the individual resources. Both assigning and sequencing of the jobs is usually subject to constraints that limit the scope of action. Typical constraints are *start time constraints* (including time windows, non-overlapping constraints and sequencing constraints) and *resource constraints* (including capacity constraints and resource qualifications). The goal of scheduling is typically to maximize profit and to reduce costs. Scheduling systems are able to create schedules that are valid with respect to the given constraints and optimal with respect to one or more goal functions. The detailed scheduling problems our guidelines address are derivatives of the *resource constrained project scheduling problem (RCPSP)*. A formal definition of the RCPSP can be found in [HB10]. We consider detailed scheduling systems with regard to their *user experience*. We define it as the extent to which the work task of the user is supported by the computer.

1.2 State of the art

In the last 20 years, many field studies have been carried out in order to evaluate the user experience of scheduling systems [CvW12, GFW11]. The results showed that scheduling software is often designed to make all decisions autonomously. It is not expected that the human scheduler participates in the process of schedule creation. Therefore it is often hard to modifiy the underlying scheduling model. However, the field studies revealed that in practice the constraints for jobs and resources and the requirements on the properties of the schedule change frequently. For each change the scheduling model has to be adapted accordingly, otherwise the resulting schedules are not applicable. Furthermore, only the human scheduler has the expert knowledge to decide at which point of time which change has to be carried out. He needs a system that supports him with finding and incorporating the right decisions. It becomes clear that there is a contradiction between the actual needs of the users and the assumption of the software industry, that full automation is sufficient. Due to this insight, the design of human computer interaction for scheduling systems has become a matter of research. Initially it was the goal to identify user actions, which allow the scheduler to incorporate his expert knowledge. A comprehensive list can be found in [dNE05]. The most important user actions deal with adapting the scheduling model (*interactive optimization*): weighting goal functions, modifying input data, changing the constraints, selecting a solution and manually modifying a solution.
It is the task of the software developers to implement the actions according to the guidelines of usability (e.g. DIN EN ISO 9241-110). They lead to interactive systems in which both human and computer take part in the solution process. However, the interaction still has some drawbacks concerning the user experience. The user actions often induce a high cognitive workload caused by the huge amount of data that is generated by the computer. For example, it is difficult for the user to compare numerous alternative solutions. In [CvW11] it is criticized: On the one side there are sophisticated algorithms to create schedules, on the other side the user is completely left alone with his actions.
For this reason the concept of function allocation between human and computer was extended from *interactive* to *mixed initiative*. As the term ”mixed” indicates, both human and computer can intervene in the solution process on their own initiative. While the user is active, the computer does not remain passive, but it helps him to successfully complete the action. This approach was applied in numerous scheduling prototypes, for example in [BZE$^+$07]. According to [BBIM96] this approach opens a new way to make the knowledge represented in the computer available for the user: The human solves a problem collaborating with the computer like with a human partner that contributes suggestions and opinions.

1.3 Contribution

In the literature there are concepts and examples of human computer interaction that can be applied to detailed scheduling systems. However, we miss a design concept that is tailored to the requirements of general detailed scheduling problems. The existing concepts have been developed either for specific problems (e.g. production scheduling) or for general optimization problems with constraints. In both cases it is difficult to reuse the interaction design in the development of a new scheduling system.
In this paper, we define an interaction concept for general detailed scheduling problems that is based on the principles of interactive optimization and mixed initiative scheduling. It helps to provide an optimal user experience in scheduling systems. Our work is structured as follows: First, we investigate the process of scheduling in practice and characterize the decision process of the scheduler (section 2). From this, we conclude requirements on the design of the human computer interaction (section 3.1). In the sections 3.2 to 3.10 we define 9 interaction guidelines that show how the requirements can be fulfilled. In the last section 4 we describe the results of a user test that confirm the effectiveness of the guidelines.

2 Human Decision and Design Processes in Scheduling

2.1 Temporary Constraints

In practice there are several typical situations in which the original scheduling model has to be updated by adding, removing or modifying the constraints. We distinguish between two different kinds of constraints: *Persistent constraints* are contained in the original scheduling model. They describe the services available to carry out the jobs (e.g. working times, capacities of resources) and the scheduling requirements of the jobs (e.g. time windows, required resources). In the remaining work we do not consider changes to persistent constraints. *Temporary constraints* represent the scheduler's decision, which resource should carry out which job at a specific point of time (cf. [Hig01]). For example, the assignment of values to decision variables (e.g. start times, resources of jobs) can be regarded as development of temporary constraints. However, we generalize the concept of temporary constraints by encompassing all kinds of constraints that narrow the domain of the variables. This allows a better modeling of human scheduling strategies. They are called ”temporary”, because they are developed during a scheduling process and discarded afterwards.

2.2 Rule-Based and Knowledge-Based Scheduling

Schedulers use templates (e.g. existing schedules from past), decision rules (e.g. if-then-statements), dispatching rules (e.g. sorting jobs by earliest start date) and other strategies

(e.g. divide and conquer) to manually solve scheduling problems. These strategies process the scheduling model in its original state, thus they can be implemented in a software. Scheduling with the help of manual strategies or arbitrary algorithms is in the following called *rule-based scheduling*.
In practice, rule-based scheduling often cannot be applied readily, because the model does not map the reality at the moment of schedule creation adequately. For example, the scheduler has expert knowledge about changed conditions, under which the resources operate (e.g. "a machine in need of repair should be used less often") or about requirements on the properties of the schedule that are not included in the model (e.g. "the schedule should be comprehensible for the staff"). The conditions and requirements can be considered by complementing the model with temporary constraints. After that, the scheduling problem can again be solved rule-based. We refer to this process as *knowledge-based scheduling*.
Often expert knowledge cannot be directly translated into temporary constraints, as it is not clear from the start, with which decisions a condition or requirement is represented best. In this case the scheduler has to set up and evaluate different constraints. The process of developing, evaluating and choosing temporary constraints can be modeled as decision process with 4 main stages (cf. [TSD11]):

Intelligence: This stage is the entry point of the decision process. Here the scheduler collects the knowledge that has to be incorporated into the schedule.
Design: In this stage alternative temporary constraints are compared and evaluated.
Choice: The final decision for a set of constraints is made in the choice stage.
Implementation: Now the selected constraints can be added to the scheduling model. A solution can be found using a rule-based procedure.

The design stage can be left out if conditions and requirements can be translated into constraints immediately.

2.3 Constraint Processing during Knowledge-Based Scheduling

In the design stage of a decision process, the scheduler evaluates alternative sets of temporary constraints expressing his preferences. Viewing scheduling as a design problem [HGM12], each set is a "‘design artifact"’ created with the following operations of constraint management [Dar91, Vis06]: *constraint posting* and *constraint propagation* [HW07].
Design problems typically are not solved by immediately posting constraints that assign a value to a variable. Instead, the domains of the variables are narrowed step by step. After each step the consequences for the remaining variables are examined by constraint propagation. The current state of the domains shows the scope for continuing the design process. In the following steps the design can be refined by posting more constraints. This procedure allows the designer to gain insights about the relationships between the variables. The effects of propagation make him question his preferences and, if necessary, change or discard them.
Building on the insights of constraint-based design, we can develop our understanding of knowledge-based scheduling further. In the design stage, temporary constraints can be posted to delay the assignment of a concrete value to a variable. For example, the

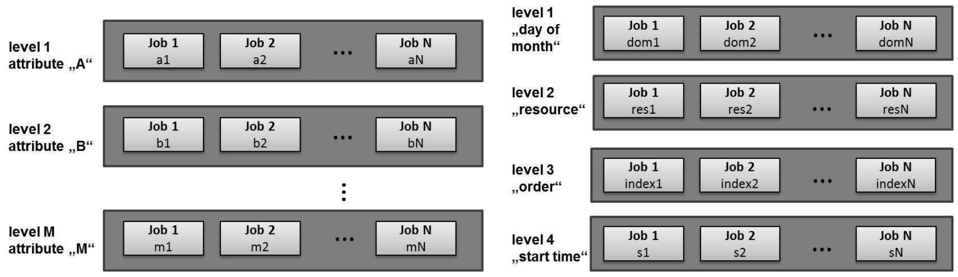


Figure 1: Template for an assignment hierarchy (left), example for fleet scheduling (right).

scheduler can define a certain time of day (e.g. morning or afternoon) before he assigns a concrete start time for a job. In doing so, an assignment for the variable ”time of day” corresponds to an *abstract assignment* of a range of values to the variable ”start time”. It could be preceded by an even more abstract assignment, for example by an assignment for the variable ”day of month”. Generally, a variable x can be used as an abstract assignment for variable y, if both variables represent attributes of the same job i and if the propagation of an assignment for x removes certain values in the domain of y. Thus, the order in which the variables of a job are assigned determines which assignment is more abstract than another. Each specific order imposes the following *assignment rules*:

1. An assignment can only be made, if all previous abstract assignments have been completed.
2. If an abstract assignment is changed, all subsequent assignments have to be renewed.

The scheduler can choose the assignment order depending on his decision situation. For example, if he is interested in the consequences of an assignment for variable x on the remaining variables, this assignment will be made first. Furthermore, assignment orders can often be derived from rule-based scheduling strategies. For example, a typical strategy to solve vehicle routing problems is to first assign each job a driver, and secondly assign each job a start time. In knowledge-based scheduling, the scheduler can maintain this strategy if it allows him to better comprehend the problem.
We suggest a concept for visualizing typical assignment orders of a scheduling problem, the *assignment hierarchy*. In this paper it will be used to illustrate the computer support for knowledge-based scheduling. Figure 1 left shows the template for an assignment hierarchy: It consists of M levels, where each level represents an attribute. Each level i contains N variables (e.g. $a_1, .., a_n$ for attribute ”A”) as a placeholder for the attribute values of all N jobs. The order of the levels corresponds to the chosen assignment order and the state of the jobs complies with the assignment rules: At level i,$i < M$ a job j is in an abstract state, as its assignments at the previous levels have already been completed, but the assignments of the subsequent levels are pending. For a scheduling problem there possibly exist multiple assignment hierarchies. Figure 1 right shows an example assignment hierarchy for a fleet scheduling problem. For each day of month each driver is assigned to a tour which defines the order and start times of the jobs to be carried out.

2.4 Factors Influencing the Effort of Knowledge-Based Scheduling

After an assignment, propagation can cause one or more variables to have empty domains or domains that do not any more contain the desired values [HW07]. In order to prevent this the scheduler would have to look ahead on the consequences on the remaining variables before assigning a value to a variable. This kind of *anticipatory propagation* is generally not possible. The reason is, first, the limited capacity of the human working memory that does not allow to process the high number of variables in practical scheduling problems [Mac08]. Second, the scheduler usually does not know the constraints of the current scheduling problem. Consequently, he is not able to perform the propagation process manually.
The difficulty of propagation leads to a blind assignment, which can end up in an empty or unsuitable domain. In the moment the scheduler notices this the responsible assignment possibly lies far back. The scheduler has to perform a backtracking process to find out and change it, which can also be very time-consuming and difficult.
Even if the scheduler finds a valid set of temporary constraints not violating any constraints, it is not guaranteed that the modified scheduling model can be solved in the implementation stage. If the scheduler does not include all variables of the model in the design stage he can assess neither the solvability nor the optimality of the considered alternative solutions. This effect makes it impossible to work with a limited set of variables, for which preferences exist, in the design stage. Furthermore, it prevents the scheduler to evaluate and compare "schedule sketches" that are composed of a few abstract assignments. Therefore we refer to this effect as *loss of abstraction*.

3 Guidelines for the Development of Interactive Scheduling Systems

3.1 Requirements on the Computer Support

We define an interaction model that provides a computer support adapted to the knowledge-based scheduling process. The requirements can be deduced from the analysis: The computer should support the design and the evaluation of alternative solutions in the design stage (sec. 2.2, 2.3). According to the mixed initiative principle, the effort caused by constraint propagation and loss of abstraction should be reduced (sec. 2.4).

3.2 Guidelines G1 to G9

3.2.1 Guideline G1 - Determining the assignment hierarchy

Prior to the development of a scheduling system, typical assignment hierarchies should be identified.
The establishment of an assignment hierarchy helps to find out, which scheduling deci-

sions and which dependencies between them are relevant for the scheduler. Furthermore, it helps to identify abstract assignments that schedulers possibly use in their heads. This is the prerequisite to create a mixed initiative computer support for the user's actions.

3.2.2 Guideline G2 - Manual modification of a schedule

G2.1: *The scheduling system should allow a manual assignment of values to all variables of the scheduling model.*
G2.2: *The scheduling system should ensure that manual assignments comply with the assignment rules of a suitable assignment order.*
If several assignment hierarchies are to be supported, the scheduler should be able to choose the current assignment hierarchy.

3.2.3 Guideline G3 - Verifying the compliance with constraints

The scheduling system should verify, whether manual assignments comply with the constraints of the scheduling model. If a constraint is violated, the scheduler should be informed.
This guideline helps the scheduler to become acquainted with the scheduling model and to refine his preferences during the design of a scheduling alternative.

3.2.4 Guideline G4 - Verifying the compliance with requirements

The scheduling system should verify, whether the schedule complies with all requirements of the organisation. After each assignment, the quality scores should be updated.
The scheduling system should supervise the assignments of the scheduler to prevent him from making decisions that significantly impair the quality scores and other requirements.

3.2.5 Guideline G5 - Explicit fixation

G5.1: *The scheduling system should allow the scheduler to define temporary constraints for all variables (fixation). They should be taken into account by automated scheduling.*
G5.2: *The fixation has to be chained upwards automatically: If an assignment is fixed at assignment level i, the assignments for the variables at the levels 1 to $i-1$ of the same job have to be fixed as well.*
With this guideline the computer can take over the rule-based implementation stage after the scheduler has defined temporary constraints. Furthermore, it helps to avoid the loss of abstraction in the design stage. In order to evaluate a scheduling alternative and its quality, the scheduler can fix his decisions at the desired assignment levels and let the computer complete the remaining assignments.
If an existing schedule is to be changed, the scheduler can fix parts of the schedule that must not be modified any more.

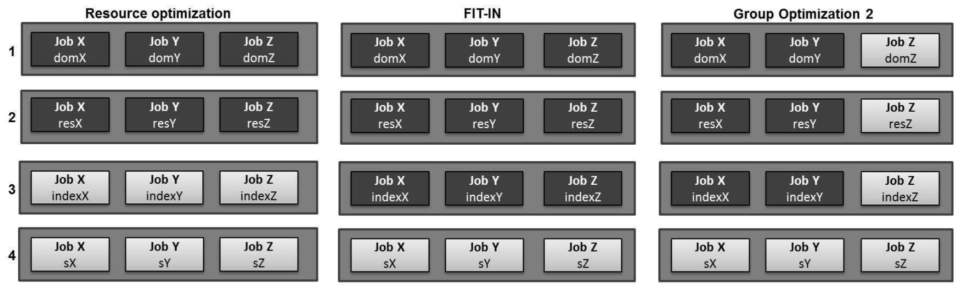


Figure 2: Templates for scheduling functions (dark grey color indicates fixed jobs).

3.2.6 Guideline G6 - Scheduling functions with implicit fixation

A scheduling system should provide scheduling functions that carry out multiple fixations and automatic scheduling in one step. They can be deduced from the assignment hierarchy of the scheduling problem.
Implicit fixation reduces the effort of manual fixation. For example, 7 different scheduling functions can be used in fleet scheduling (fixed levels/jobs are given in brackets, see figure 2): *resource optimization* (level 1 and 2), *resource optimization 2* for optimizing a tour (levels 1 and 2 for a selected resource, levels 1 to 4 for the rest of the schedule), *FIT-IN* for repairing a schedule with overlapping jobs (levels 1 to 3), *group optimization* for rescheduling selected jobs (levels 1 to 4 for non-selected jobs), *group optimization 2* where non-selected jobs can be shifted in time (level 1 to 3 for non-selected jobs) and *full optimization* where all jobs are scheduled (no fixation).

3.2.7 Guideline G7 - Support for value selection

The scheduling system should support the scheduler in selecting a value from the domain of a variable. The support can be implemented in three different ways:
G7.1: *The scheduling system suggests values that lie in the domain of the variable.*
G7.2: *After each assignment, the scheduling system removes all values from the domains of the unassigned variables that cause a conflict with the assigned variables. It suggests values that lie in the reduced domain of the variable.*
G7.3: *After each assignment, the scheduling system removes all values from the domains of the unassigned variables that are not part of a solution extending the current partial schedule. It suggests values that lie in the reduced domain of the variable.*
Guideline G7.1 prevents the scheduler from using a trial-and-error strategy for value selection, if he doesn't know the domain of a variable. Guideline G7.2 is useful if an assignment has to be changed in an existing schedule: The computer only suggests values that do not cause the violation of constraints. With guideline G7.3 the computer takes over the propagation of assignments. Values that will cause empty domains in future assignment steps are removed from the domain of the considered variable. If the scheduler selects one of the suggested values, he can be sure, that no backtracking will be necessary later. This way the

effort of knowledge-based scheduling is reduced. Furthermore, G7.3 allows the scheduler to delay the automatic completion of the schedule until the implementation stage, in which the temporary constraints are fixed (G5).
This guideline can be implemented with algorithms that achieve k-consistency with $k \geq 2, k \leq N$ for a constraint optimization problem with N variables [HW07]. The higher k the more values can be removed from the domains but the higher is the required computing time. For this reason, in practice it can be required to make a compromise, where as many values as possible are removed in an acceptable time.

3.2.8 Guideline G8 - Weak fixation

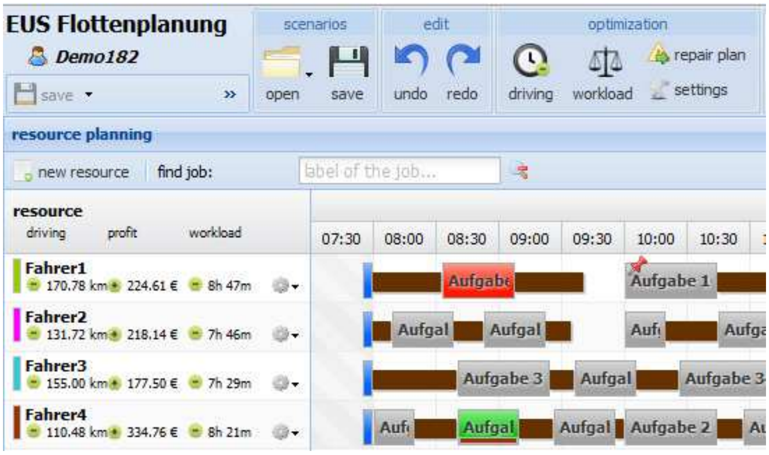
The scheduling system should allow the scheduler to define temporary constraints from which the computer can deviate during automated scheduling if the problem is otherwise not solvable (weak fixation). Thereby, the deviation from the original assignments should be as small as possible.
The weak fixation can be used to find a set of temporary constraints, for which the scheduling problem is solvable. If the domain of a variable does not contain the desired value any more because it was removed during the propagation of previous assignments, the scheduler has two options: He could fix the desired assignment of this variable normally and the assignments of the previously considered variables weakly. This way the values of the latter variables are adapted during automatic scheduling. Alternatively he could fix the values of all variables weakly and let the computer find a compromise with a small deviation. In both cases the scheduler is relieved from the effort of manual backtracking.

3.2.9 Guideline G9 - Extended support for value selection

If the scheduler wants to assign an value from the domain of a variable that is inconsistent with the current state of the schedule, the scheduling system should determine the assignment level, on which the schedule has to be changed in order to enable this assignment.
Guideline G9 helps to assess the extent to which an existing schedule has to be changed in order to enable an assignment currently violating one or more constraints. For example, consider a fleet scheduling problem where the scheduler wants to assign a job to a different resource. Three different situations can be distinguished, if there is no gap in the tour of this resource, where the job can be inserted:

- *Change on level 4:* In the tour of this resource there is a gap which can be widened by adjusting the start times of the jobs. However, the existing order of the jobs can remain unchanged.
- *Change on level 3:* The order of the jobs in the tour of this resource has to be changed before the new job can be inserted.
- *Change on level 2:* One or more jobs have to be removed from the tour before the new job can be inserted.

The required changes might influence the preference of the scheduler. If they are too big, the scheduler possibly does without the assignment.



resource	driving	profit	workload
Fahrer1	170.78 km	224.61 €	8h 47m
Fahrer2	131.72 km	218.14 €	7h 46m
Fahrer3	155.00 km	177.50 €	7h 29m
Fahrer4	110.48 km	334.76 €	8h 21m

Figure 3: User interface for fleet scheduling.

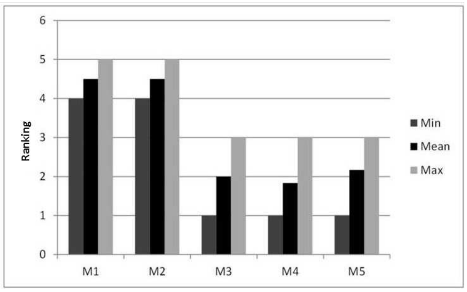
4 Evaluation of the Guidelines

4.1 Test hypothesis

Conventional scheduling systems often provide one of these interaction models: *manual scheduling* (M1), *full automatic scheduling with subsequent manual modification of the schedule* (M2), *provision of alternative schedules by the computer*. The latter model is not considered in the test, as it doesn't allow knowledge-based scheduling. Models M1 and M2 allow knowledge-based scheduling, but they do not include computer support to reduce the effort (see section 2.4). We assume that depending on the problem structure, it can be difficult to incorporate temporary constraints, such that the resulting schedule is both valid (no constraints are violated) and optimal. Our hypothesis is that an interaction model based on our guidelines provides a better user experience to achieve these properties.

4.2 Test setup and results

We have carried out a user test in order to prove our test hypothesis. In this test, 45 participants (students from our university) solved 6 typical scheduling tasks in the application area of fleet scheduling. They had to change the resource assignments, the order and the start times of certain jobs in complete schedules or they had to create new schedules incorporating similar preferences. Constraints to consider were start time constraints (time windows for jobs) and resource constraints (limited choice of resources for some jobs). Furthermore, the participants should try to achieve a good quality score.
The participants were provided with a scheduling system, whose user interface is depicted in figure 3. The system was configured in 5 different test models M1 to M5. Test models



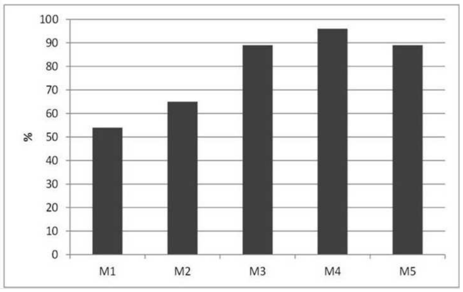


Figure 4: Ranking of the models with regard to the quality (left) and average task success (right).

M1 and M2 correspond to the interaction models M1 and M2 in section 4.1. Test models M3 to M5 represent our new interaction model in 3 different ways: In test model M3 the system provides fixation features at the levels 2 and 4[1]. The test models M4 and M5 additionally provided the scheduling functions *resource optimization 2, group optimization 1/2*, and *FIT-IN*. In order to encourage the use of the scheduling functions (see figure 2) *full optimization* was missing in M4. In all models a minimal support for manual scheduling was provided according to the guidelines G1 to G5 and G7.2[2]
Each scheduling task was solved using each model which allows us to compare the effectiveness of the models with regard to the metrics quality and task success. Figure 4 shows the overall ranking of the models with regard to the achieved quality scores (1 is the best, 6 the worst rank). The ranks were averaged over the six scheduling tasks. Models 3, 4 and 5 produce the shortest travel time. The worse quality achieved with M1 and M2 can be attributed to the high effort of manual scheduling. The participants often had to backtrack their assignments. Furthermore, in their effort to fulfill all constraints they were not able to simultaneously strive for a good quality.
A task was completed successfully, if all jobs were scheduled according to the given preferences and no constraints were violated. Figure 4 shows the distribution of the average success rate over all 5 models. The success rate is not more than 60%, if manual scheduling is required (M1, M2). The high success rate of the models M3 to M5 confirms our assumption that manual scheduling generally is inconvenient for humans.

4.3 Conclusions

The results show that the interaction models M1 and M2 are not sufficient for knowledge-based scheduling. They confirm our hypothesis that a high quality and task success rate can only be achieved if fixation facilities or scheduling functions at suitable assignment levels are provided.
As the participants were not scheduling experts, the preferences had to be given in the tasks. They did not carry out a search for constraints in the design stage, thus the guidelines

[1]Fixation at levels 1 and 3 was not necessary to perform the tasks.
[2]In a previous test nearly all participants failed if no support for value selection or only G7.1 was provided.

G7.3, G8 and G9 could not be evaluated. This has to be done in a field study.

References

[BBIM96] Mark H. Burstein, Bolt Beranek, Newman Inc, and Drew V. Mcdermott. Issues in the development of human-computer mixed-initiative planning. In *Cognitive Technology*, pages 285–303. Elsevier, 1996.

[BZE+07] Keith A. Butler, Jiajie Zhang, Chris Esposito, Ali Bahrami, Ron Hebron, and David E. Kieras. Work-centered design: a case study of a mixed-initiative scheduler. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 747–756, 2007.

[CvW11] Julien Cegarra and Wout van Wezel. Allocating Functions to Human and Algorithm in Scheduling. In Jan C. Fransoo, Toni Waefler, and John R. Wilson, editors, *Behavioral Operations in Planning and Scheduling*, pages 339–370. Springer Berlin Heidelberg, 2011.

[CvW12] Julien Cegarra and Wout van Wezel. Revisiting Decision Support Systems for Cognitive Readiness: A Contribution to Unstructured and Complex Scheduling Situations. *Journal of Cognitive Engineering and Decision Making*, 6(3):299–324, 2012.

[Dar91] Francoise Darses. The constraint satisfaction approach to design: A psychological investigation. *Acta Psychologica*, 78(1-3):307–325, 1991.

[dNE05] Hugo A.D. do Nascimento and Peter Eades. User hints: a framework for interactive optimization. *Future Generation Computer Systems*, 21(7):1177–1191, 2005.

[GFW11] Roland Gasser, Katrin Fischer, and Toni Wäfler. Decision Making in Planning and Scheduling: A Field Study of Planning Behaviour in Manufacturing. In Jan C. Fransoo, Toni Wäfler, and John R. Wilson, editors, *Behavioral Operations in Planning and Scheduling*, pages 11–30. Springer Berlin Heidelberg, 2011.

[HB10] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.

[HGM12] Jean-Michel Hoc, Clément Guerin, and Nasser Mebarki. The nature of expertise in scheduling: The case of timetabling. *Human Factors and Ergonomics in Manufacturing and Service Industries*, 2012.

[Hig01] Peter G. Higgins. Architecture and interface aspects of scheduling decision support. In *Human performance in planning and scheduling*, pages 245–280. Taylor & Francis, 2001.

[HW07] P. Hofstedt and A. Wolf. *Einführung in die Constraint-Programmierung: Grundlagen, Methoden, Sprachen, Anwendungen*. Springer, 2007.

[Mac08] M. MacDonald. *Your Brain: The Missing Manual*. Missing manual. O'Reilly Media, 2008.

[SFG+12] Hartmut Staedler, Bernhard Fleischmann, Martin Grunow, Herbert Meyr, and Christopher Sürie. *Advanced Planning in Supply Chains - Illustrating the Concepts Using an SAP APO Case Study*. Springer Berlin Heidelberg, 2012.

[TSD11] Efraim Turban, Ramesh Sharda, and Dursun Delen. *Decision Support and Business Intelligence Systems*. Pearson, 2011.

[Vis06] Willemien Visser. Designing as construction of representations: a dynamic viewpoint in cognitive design research. *Human-Computer Interaction*, 21(1):103–152, 2006.