

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Master's thesis



Martina Kopecká

**Methods of interaction with advanced planning and
scheduling systems for production**

Study program: Open Informatics
Field of study: Human-Computer Interaction
Thesis supervisor: Ing. Antonín Novák, Ph.D.

2023

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kopecká** Jméno: **Martina** Osobní číslo: **487030**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávací katedra/ústav: **Katedra počítačové grafiky a interakce**
Studijní program: **Otevřená informatika**
Specializace: **Interakce člověka s počítačem**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Metody interakce s pokročilými plánovacími a rozvrhovacími systémy pro výrobu

Název diplomové práce anglicky:

Methods of interaction with advanced planning and scheduling systems for production

Pokyny pro vypracování:

Moderní plánovací a rozvrhovací systémy (tzv. APS) jsou nástroje, které plánovačům pomáhají navrhnout rozvrh a organizaci výroby zakázek. Reálné použití těchto nástrojů ukazuje, že jednoduché Ganttovy diagramy neposkytují plánovači dostatečný přehled o parametrech a konsekvencích navrhovaného rozvrhu výroby tak, aby mohl s rozvrhovacím algoritmem interagovat a dodatečně specifikovat další omezení úlohy. Cílem této práce je navrhnout, vytvořit a vyhodnotit nové metody interakce s pokročilými plánovacími a rozvrhovacími systémy. Těžiště spočívá v identifikaci potřeb profesionálních uživatelů APS a v navržení kreativních metod vizualizace a interakce pro práci s navrženými rozvrhy.

- 1) Seznamte se se stávajícím stavem uživatelských rozhraní a metod interakce s plánovacími a rozvrhovacími systémy pro výrobu.
- 2) Vytvořte základní prototyp APS spolu s optimalizačním algoritmem pro zjednodušený model výroby chápáný jako rozšířený problém rozvrhování projektů s kalendářní zdrojů.
- 3) Vyhodnoťte uživatelským testováním vhodnost/limitace základních interakčních a vizualizačních komponent nalezeného rozvrhu s ohledem na potřeby profesionálních uživatelů APS.
- 4) Navrhněte a implementujte nové metody vizualizace a interakce pro zvolené komponenty APS.
- 5) Otestujte navržené řešení interakce a vizualizace s profesionálními uživateli APS systémů.

Seznam doporučené literatury:

- [1] Goodman, E., Kuniavsky, M. & Moed, A. (2012). Observing the user experience: A practitioner's guide to user research. Elsevier.
- [2] Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About face: the essentials of interaction design. John Wiley & Sons.
- [3] Pinedo, Michael L. Scheduling: Theory, Algorithms, and Systems. Vol. 29. New York: Springer, 2012.
- [4] Framinan, J. M., Leisten, R., & García, R. R. (2014). Manufacturing scheduling systems. An integrated view on Models, Methods and Tools, 51-63.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Antonín Novák, Ph.D. katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **09.02.2023**

Termín odevzdání diplomové práce: **26.05.2023**

Platnost zadání diplomové práce: **22.09.2024**

Ing. Antonín Novák, Ph.D.
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomantka bere na vědomí, že je povinna vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání

Podpis studentky

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Prague, May 2023

.....
Martina Kopecká

Acknowledgements

I would like to express my gratitude to my supervisor Antonín Novák for his guidance and support. I would also like to thank prof. Zdeněk Hanzálek for sharing his invaluable insights into scheduling.

Special thanks go to the experts participating in the user testing.

Last but not least, I would like to thank Dan Justiz for his help with the revision.

Abstract

This thesis examines the problem of interaction in Advanced Planning and Scheduling (APS) systems. Even though the role of human planners has proven essential, the support they receive for their tasks, such as improving a production schedule, remains limited in state-of-the-art applications. The typical representations of schedules, such as Gantt charts, are inappropriate for complex problems, mainly due to their limited scalability. Still, adequate graphical representations can improve understanding. Employing human-centered design principles, we build a prototype user interface featuring several novel visualization components, such as advanced capacity analysis, which help planners with decision-making. To tackle the problem of comparing multiple schedules, we present a versioning system similar to Git, which professional users of APS systems perceived positively. We discovered users could effectively manipulate the schedules by adjusting the problem models.

Keywords

Advanced Planning and Scheduling systems, human-centered design, user interface, production scheduling, exploratory user testing

Abstrakt

Tato diplomová práce pojednává o problematice interakce v pokročilých plánovacích a rozvrhovacích systémech (tzv. APS). Ačkoli se ukazuje, že role uživatelů-plánovačů je nezastupitelná, podpora pro úlohy, které řeší, například upravování rozvrhu výroby, je ve stávajících aplikacích omezená. Běžné způsoby reprezentace rozvrhů, jakými jsou například Ganttovy diagramy, jsou pro složité úlohy nevhodné, a to zejména kvůli omezené škálovatelnosti. Vhodné grafické reprezentace přesto mohou podporovat porozumění. Za pomoci principů designu zaměřeného na člověka navrhujeme prototyp uživatelského rozhraní, které obsahuje několik nových vizualizačních prvků, jako například pokročilou kapacitní analýzu, které plánovačům pomáhají s rozhodováním. Abychom vyřešili problém porovnávání více rozvrhů, navrhujeme verzovací systém podobný nástroji Git, který profesionální uživatelé systémů APS dobře přijali. Zjistili jsme, že uživatelé mohou s rozvrhem efektivně manipulovat pomocí úprav modelu úlohy.

Klíčová slova

pokročilé plánovací a rozvrhovací systémy, design zaměřený na člověka, uživatelské rozhraní, rozvrhování výroby, exploratorní uživatelské testování

List of Abbreviations

Abbreviation	Meaning
API	application programming interface
APS	advanced planning and scheduling
ERP	enterprise resource planning
HTA	hierarchical task analysis
IS	information system
KPI	key performance indicator
MRP	material resource planning
PSPLib	Project Scheduling Problem Library
RCPSP	resource-constrained project scheduling problem
UI	user interface
UX	user experience
VCS	version control system

Contents

1	Introduction	1
1.1	Research Question	1
1.2	Research Objectives	2
1.3	Thesis Outline	2
I	Analysis	3
2	Related Work	5
2.1	Decision-Making Behavior	5
2.2	Interactive Scheduling Systems	6
2.2.1	Design Model for Scheduling Systems	6
2.2.2	Expert System Approach	6
2.2.3	Interactive Optimization Approach	6
2.3	Designing for Complex Domains	7
3	Methodology	9
3.1	Involvement of Domain Experts and Users	9
3.2	Prototyping	9
3.3	User Testing	10
4	Planning and Scheduling Systems	13
4.1	Advanced Planning and Scheduling	13
4.2	Functionalities	14
4.3	Architecture of a Scheduling System	14
4.4	User Interfaces of Scheduling Systems	15
4.4.1	Gantt Chart Interface	15
4.4.2	Capacity Buckets Interface	16
4.4.3	Data Table Interfaces	16
4.5	User Interfaces in Commercial Systems	17
5	Design Problem	21
5.1	Problem Statement	21
5.2	Persona Hypothesis	21
5.2.1	Primary persona	22
5.2.2	Non-user persona	22
5.3	Scenarios	23
5.4	Hierarchical Task Analysis	23
5.5	Design Requirements	24
II	Prototype	27
6	Model of Scheduling Problem	29
6.1	Resource Constrained Project Scheduling Problem	29

6.2	Problem Modifications	30
6.3	Constraint Programming Model	31
7	Solution Prototype I	33
7.1	Key Concepts	33
7.2	Prototype Technologies	34
7.3	UI Components	34
7.3.1	Orders Table and Activities Table	35
7.3.2	Solution Comparison	35
7.3.3	Capacity Analysis	36
7.3.4	Pairwise Comparison	36
7.3.5	Changes	37
7.3.6	Resource Calendar	37
7.3.7	Gantt Chart Visualization	37
7.3.8	Version Tree	38
8	Prototype I Evaluation	39
8.1	Methodology	39
8.2	Results	40
8.3	Discussion	41
9	Solution Prototype II	43
9.1	Ideation	43
9.2	Clickable Mockup	44
9.3	Clickable Mockup Testing	45
9.4	Coded Prototype	46
9.4.1	Orders Table and Activities Table	46
9.4.2	Resource Calendar	47
9.4.3	Capacity Analysis	48
9.4.4	Solution Comparison and Version Tree	48
9.4.5	Resource Activities	49
9.4.6	Order Analytics	49
10	Prototype II Evaluation	51
10.1	Methodology	51
10.2	Results	51
10.3	Discussion	53
11	Conclusion and future work	55
11.1	Fulfillment of the Objectives	56
11.2	Future Work	56
	Appendix	61
A	Design Process Documentation	61
B	User Testing Documentation	61
C	Source Code	61

List of Figures

2.1	Human decision-making model	5
2.2	Problem-solving loop	7
3.1	Wizard of the Oz prototype	10
4.1	Configuration of a scheduling system	15
4.2	Gantt chart interface	16
4.3	Capacity buckets interface	16
4.4	Data table interface	17
4.5	Dispatch list interface	17
4.6	Overview of orders	18
4.7	Capacity balance	18
4.8	Gantt chart	19
4.9	List of products and Gantt chart	19
4.10	List of operations on a resource	20
5.1	Avatar of primary persona	22
5.2	Avatar of non-user persona	22
5.3	Hierarchical Task Analysis diagram	24
6.1	Visualization of RCPSP definition	29
6.2	Visualization of RCPSP solution	30
7.1	Information architecture	34
7.2	The main page screenshot	35
7.3	Pairwise comparison window	36
7.4	Main page with changes panel open	37
7.5	Resource calendar	37
7.6	Gantt chart visualization	38
7.7	Version tree	38
9.1	Sketches of modified capacity analysis	43
9.2	Redesigned main interface	44
9.3	The main interface of the clickable mockup	44
9.4	Modified capacity analysis interface	45
9.5	Information architecture	47
9.6	The main page screenshot	47
9.7	Resource calendar	47
9.8	Resource in capacity analysis interface	48
9.9	Resource activities screenshot	49
9.10	Order analytics screenshot	50
9.11	Duration x Potential scatterplot	50

List of Tables

8.1	Profiles of first coded prototype testers	39
9.1	Profiles of clickable mockup testers	45
10.1	Profiles of second coded prototype testers	51

1 Introduction

In scheduling theory, the problems are usually well-defined, and the objectives and parameters, or at least their probability distributions, are known in advance. In contrast, real-world planning and scheduling situations often have the traits of so-called wicked problems named by Rittel and Webber [1], i.e.,

- the problem is ill-defined,
- the problem has no stopping rule—the planner stops for considerations that are external to the problem, e.g., when he runs out of time, money, or patience,
- the solutions are not true-or-false but good-or-bad,
- the set of potential solutions is not enumerable,
- the planner has no right to be wrong.

Therefore, bridging the gap between theory and practice and deploying advanced planning and scheduling methods in real-world manufacturing environments can be challenging. Even in recent years, spreadsheet applications have played a major role in production planning and scheduling and still dominate over Advanced Planning and Scheduling (APS) systems [2].

The *APS systems* are software tools that utilize complex mathematical algorithms to support decision-making about future production [3]. No matter how advanced the tool may be, sometimes user intervention is necessary. The schedule presented by the system—which is often not even optimal due to the limited execution time of optimization procedures—may not satisfy the user’s requirements. For instance, he can reject a solution for some implicit reasons or accept a solution that violates certain constraints [4].

Practice shows that people are superior to computers in some aspects of scheduling. Specifically, McKay and Wiers [5] state that, unlike computers, humans can:

- cope with goals and constraints that may be implicit, incomplete, or erroneous,
- communicate and negotiate with operators on the shop floor, customers, or suppliers,
- fill in the missing information required for scheduling.

Yet state-of-the-art interfaces for interaction between the human planner and the scheduling system typically incorporate Gantt charts—i.e., horizontal bar charts in which the x -axis represents time, the y -axis represents the different machines or stages in the manufacturing process and jobs or activities are depicted as rectangles with basis corresponding to their processing time [6, p. 323]. While the concept of Gantt charts is relatively easy-to-understand, such visualizations do not provide sufficient support for the scheduling process in real manufacturing environments due to the limited scalability, explorability, or reschedulability [7].

1.1 Research Question

As we mentioned earlier, state-of-the-art scheduling systems do not adequately support the planners. Due to that, the effective forms of human-computer interaction in scheduling

1 Introduction

systems are an opportunity for further research. Our aim for this work was to investigate *how can the user (i.e., a domain expert in scheduling) manipulate a schedule and keep track of situation*. We refined this aim into two main research questions:

- (i) *What methods of interaction are suitable for manipulation of schedules?*
This question aims to identify the methods of interaction that enable manipulating the schedule.
- (ii) *How to present the information about schedules to the users?*
This question seeks to determine what schedule-related information is valuable and how it should be presented to the users.

1.2 Research Objectives

Our objectives for this work are following:

- Analyze the forms user interfaces (UI) of APS systems take.
- Create a first prototype with typical components of APS system.
- Test the first prototype with experts in scheduling and identify their needs.
- Design novel visualization components and interaction methods and implement them into the prototype.
- Evaluate the prototype with professional users of APS systems.

1.3 Thesis Outline

This thesis consists of two parts (Analysis and Prototype) and contains total of 11 chapters.

The first part is dedicated to problem analysis. In Chapter 2, we survey the previous research on human factors in scheduling. In Chapter 3, we introduce the methodology we used for designing the interface. In Chapter 4, we describe the APS systems, including their functionalities, their conceptual architecture and the forms their UIs take. In Chapter 5, we formulate the design problem using several user experience (UX) design methods.

The second part is dedicated to the development of a solution prototype. In Chapter 6, we define the model of a scheduling problem we used for simulating the problem-solving capabilities of the APS system. In Chapter 7, we introduce the first version of the prototype. In Chapter 8, we describe the process of its evaluation. In Chapter 9, we describe the second coded prototype and the design process toward it. In Chapter 10, we explain the process of its evaluation.

In Chapter 11, we present the conclusions from the findings and outline the directions for future work.

Part I
Analysis

2 Related Work

The research on human factors in scheduling is not novel. Decision-making behaviors of human planners and the concept of interactive production scheduling systems have been studied. Although previous research provides a broad overview of the tasks complex decision-making software should assist with, the interaction design of APS systems has been neglected.

2.1 Decision-Making Behavior

Naturalistic decision-making attempts to understand how people, typically domain experts, make decisions under challenging conditions such as limited time, uncertainty, high stakes, or vague goals [8]. Wiers [9] adapted naturalistic models of human decision-making to describe scheduling situations (Figure 2.1). The distinction between skill-, rule- and knowledge-based behaviors was introduced by Rasmussen [10]. According to his model, some problems are solved without conscious control (skill-based level) or with explicit know-how (rule-based level), while others require the development of a new plan and its testing against the goal, possibly by trial-and-error (knowledge-based level). This model gives a broad overview of the cognitive tasks the UI of APS systems should assist with.

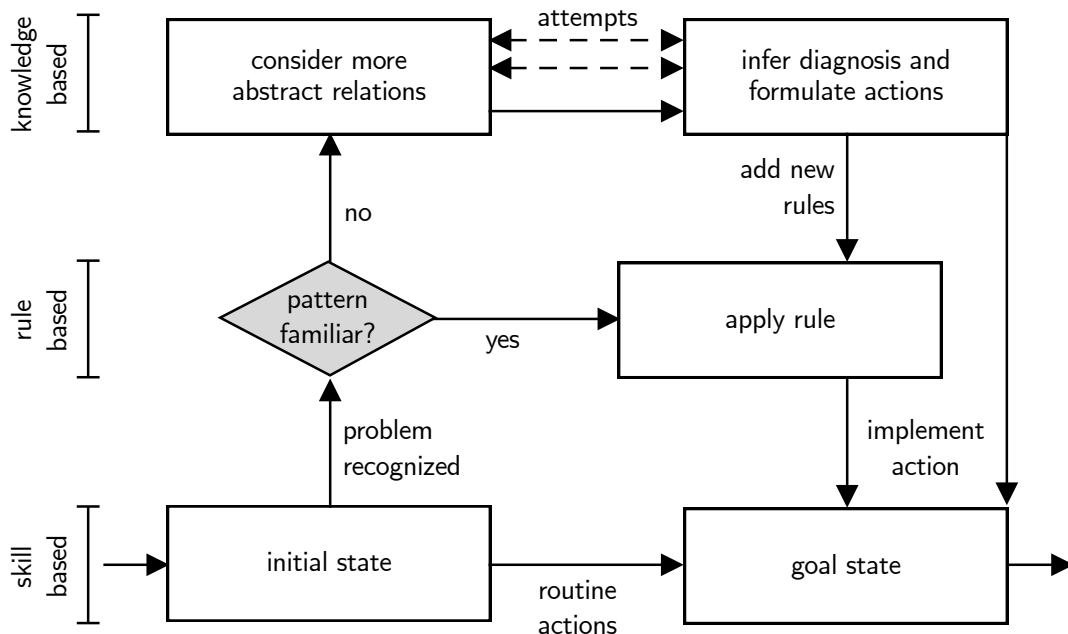


Figure 2.1: Human decision-making behavior model. Adapted from Wiers [9].

Cegarra [11] attempted to bring laboratory and field studies of human behavior in scheduling situations closer together using a cognitive typology. This cognitive typology associates seven dimensions—(i) complexity, (ii) uncertainty of information and the future state of production, (iii) time pressure, (iv) cycle synchronicity, (v) process steadiness,

(vi) process continuity, and (vii) multiple and contradictory objectives—with their related human strategies and with the implications for scheduling tool design.

For instance, planners manage the *complexity* by categorizing orders, detecting known situations, or relaxing constraints. Hence, the interfaces of scheduling tools should support the identification of situation patterns and allow the user to decide on the flexibility of the constraints [11].

2.2 Interactive Scheduling Systems

The literature describes two approaches to incorporating human knowledge into the scheduling process. In the first approach (expert system approach), the system simulates the behaviors of human planners. In the latter (interactive optimization), the user iteratively evaluates the solutions and adjusts the problem parameters.

2.2.1 Design Model for Scheduling Systems

Wiers [9] studied why the scheduling systems are (not) used in practice and how human planners can be supported by scheduling systems. He conducted four case studies and introduced four concepts important for analyzing and designing scheduling systems.

Autonomy indicates whether shop floor operators are allowed to perform certain corrective actions when disturbances occur, or the reaction of a planner is required.

Transparency of the scheduling system influences the extent to which the planner feels he is in direct control of the scheduling procedure.

Level of support for schedule generation is related to sharing responsibilities between human planners and scheduling systems.

Information aggregation is necessary due to the complexity of scheduling problems. Different levels of information aggregation should be supported.

However, he did not finish the implementation of the decision support system, and the model was not evaluated in practice.

2.2.2 Expert System Approach

The idea behind the expert system approach is simple—the scheduling system should simulate the behaviors of human planners. Kerr and Ebsary [12] found out that the employee responsible for manual scheduling in a small company had a large amount of knowledge and a set of apparently successful informal rules in mind. They tried to simulate his behavior but discovered that it was difficult to capture a changing knowledge base in a highly dynamic environment. Fox [13] shows another problem with this approach:

Problems like factory scheduling tend to be so complex that they are beyond the cognitive capabilities of the human scheduler. Therefore, the schedules produced by the scheduler are poor; nobody wants to emulate their performance.

2.2.3 Interactive Optimization Approach

In interactive optimization, the user is involved in the optimization process and can change the result or performance of the optimization (*human-in-the-loop* approach) [14]. These methods have already been applied in commercial APS software.

Liu et al. [15] provide a theoretical framework (*problem-solving loop*, see Figure 2.2) for understanding the high-level user goals and processes in interactive optimization. There are two main loops—the *model-definition loop* captures the development of the mathematical optimization model, whereas the *optimization loop* captures the use of the model by the end-user and the support for decision-making.

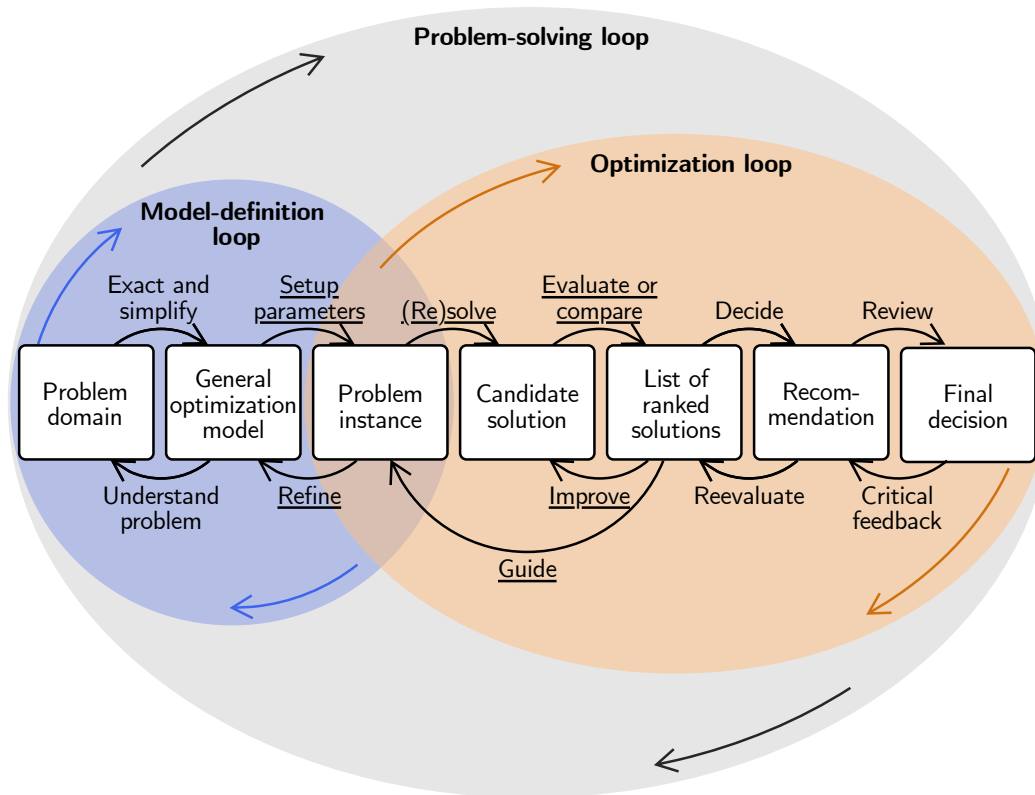


Figure 2.2: Problem-solving loop. Tasks that should be supported by the interactive tool are underlined. Adapted from Liu et al. [15].

Liu et al. [16] list several recommendations on designing interactive optimization systems. According to their suggestions, the system should:

- provide appropriate visual representations of solutions and constraints,
- support user modifications of the optimization model,
- allow direct modification of solutions,
- support comparison of solutions,
- record solution provenance.

2.3 Designing for Complex Domains

APS systems are an example of so-called *complex applications for specialized domains*. Unlike *generalist applications for everyday domains*, which “enable non-specialized users to complete largely discrete, linear tasks organized around well-structured goals”, complex applications require expertise and support broad, unstructured goals or non-linear workflows [17].

Typically, there are many sources of complexity, including the volume of data and the need to analyze it (information complexity), unstructured goals and broad tasks (intention

2 *Related Work*

complexity), or competing environmental elements, interruptions, and distractions for users (environmental complexity) [17].

Visual analytics, i.e., combining automated analysis techniques with interactive visualizations, seem promising for tackling the information complexity [18, p. 7]. This concept appears applicable even for the design of APS systems. Unfortunately, it is not a mature technology, and no guidelines for designing such applications exist [18, p. 147].

Kaplan [19] provides general recommendations, such as incorporating domain experts throughout the design process or adapting traditional UX methods if the designers struggle to apply them. Nathan [20] states that the experts can tolerate bad UI but definitely will not tolerate bad content. In his experience, the experts are particularly change-averse and distrust the tools that claim they have anywhere near the expertise that the experts do.

3 Methodology

Our methodology is built upon the basic principles of *human-centered design*—an approach to the design of interactive systems that puts focus on the users and their needs or requirements while applying human factors and usability knowledge and techniques [21]. We aim to understand the needs of APS system users and generate novel design ideas based on that. The design process is iterative, with three main pillars:

- (i) involvement of users and domain experts,
- (ii) prototyping, and
- (iii) exploratory user testing.

3.1 Involvement of Domain Experts and Users

We cooperated with the experts throughout the whole design process. User research was not just a single event taking place in the beginning. Instead, we adopted the *continuous discovery principle* from Lean UX framework—the research activities are built into every sprint [22, pp. 74–76]. We recruited the users using criteria described in Section 3.3. In the early phase, we informally presented our ideas to the experts. Later on, we asked the users to test the prototypes and provide feedback.

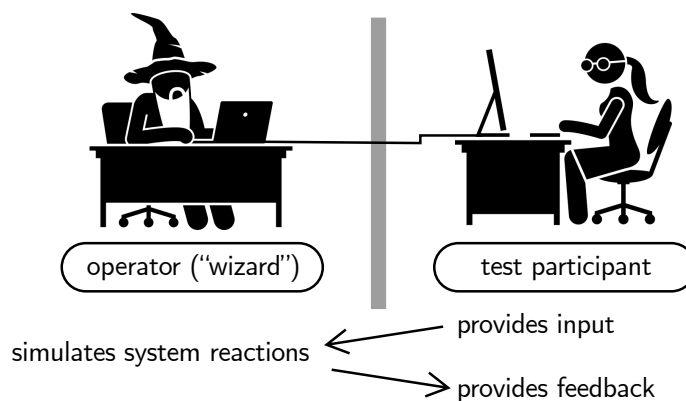
This approach helped us understand the complex domain and uncover the needs and expectations the users have.

3.2 Prototyping

A prototype is an approximation of an experience with the product in question. Software prototypes are typically distinguished in terms of their fidelity, *high-fidelity* prototypes are more similar to the final product. In contrast, *low-fidelity* prototypes differ from the final product in visual appearance, interaction style, or level of detail and are more sketchy. Other differences between prototypes arise from the choice of medium, which, in the case of software applications, may be either paper or computer [23]. The choice of the correct medium and level of fidelity depends on the audience, the objectives of the prototype, and the time available to create it [22, p. 58].

Our primary objective was to design an interactive tool that supports planners in analyzing a schedule and improving the problem model. We focused on this part of the workflow—on which the most user effort is expended—rather than on the overall experience with the APS system. Rubin and Chisnell [24, p. 31] state that the prototype should show enough functionality to address the particular test objective. There exist methods for simulating system responses without coding, such as Wizard of the Oz prototyping (Figure 3.1) [25]. However, scheduling system responds with schedules and analytical data—which are harder to mock than to code. Therefore, the first prototype we created and tested with users was an interactive *coded prototype*.

The second type of prototype we created was a *clickable mockup*, which we used to validate new visualizations and the information architecture.

Figure 3.1: Wizard of the Oz prototype¹.

3.3 User Testing

The purpose of the prototype is to test the design ideas. Goodman and Kuniavsky [26, p. 274] distinguish four types of usability tests: (i) *exploratory* to test preliminary concepts and evaluate their promise, (ii) *assessment* to test features during implementation, (iii) *comparison* to assess one design against another, and (iv) *validation* to certify that features meet certain standards and benchmarks late in the development process.

Given that the main objective of prototyping was to design a tool that supports users in modeling a scheduling problem, testing aimed to evaluate the promise of the design ideas to such support. The testing was based on solving a scheduling problem described in Chapter 6 in the interface.

We used qualitative methods of evaluation. The first method was an *asynchronous testing* allowing the users to test on their own. We got their feedback from self-reports and a group interview we held afterward. The second method we used was traditional *in-person moderated testing*. The users were asked to verbalize their thoughts (so-called *think-aloud protocol* [24, p. 54]) while interacting with the scheduling system, analyzing the situation, and proposing strategies for model improvement. In case they were stuck, the moderator provided them with clues.

The tests have to be conducted with relevant users to provide valid information. In general, two factors of user recruitment shall be considered—the representativeness of the sample and the size of the sample. The representativeness of users can be determined by age, gender, education, job responsibility or domain expertise, technical experience, and experience with specific software or hardware [27, p. 274]. Regarding the sample size, Virzi [28] states “80% of the usability problems are detected with four or five subjects”. However, the optimal number of participants should be given by the available resources such as money, time, or number of participants willing to help [27, p. 276].

In this particular case, the evaluators should have a background in the field of production scheduling. Hence we considered the *domain expertise* in production scheduling and *professional experience with APS system* significant—we wanted to recruit the users who were already familiar with some APS system rather than novices. However, this significantly limits the number of users that can be recruited. Moreover, the experts often have limited time available. Due to that, it is often necessary to violate the best practices in user testing. A typical workaround is to replace the experts with undergraduate students and

¹Icons from <https://flaticon.com>.

have them evaluate the tool; yet it is unclear to what extent the results found carry to real users [18, p. 133].

We ought to get domain-specific feedback, and we were able to recruit several experts on scheduling experienced with APS systems. However, their number was insufficient to test the interface with a new set of participants for every study, so we reused them in multiple iterations. We also conducted one study asynchronously because of their busy schedules.

Nevertheless, it is rather challenging to evaluate the support the tool gives for decision-making—for example, it is not always possible to trace if the user made the discovery through the use of a visualization [29]. While the methodologies for evaluating such tools remain an open research problem, qualitative studies appear to be particularly suitable [18, p. 136]. The scenarios the users solve during the testing should simulate the situations in which they would be using the system [26, pp. 283–285]. In this case, we acted like the managers and named them several priority orders and the desired state, but otherwise, we let anything up to them—like in the real-world setting. However, the set of possible solutions is not enumerable, and it is hard to determine an optimal flow to compare their performance with.

If we discover any usability issues, we rate their severity using the terminology and scale of Nielsen [30]:

- *cosmetic problem only*—need not be fixed unless extra time is available on the project,
- *minor usability problem*—fixing this should be given low priority,
- *major usability problem*—important to fix, so should be given high priority,
- *usability catastrophe*—imperative to fix this before the product can be released.

4 Planning and Scheduling Systems

A planning and scheduling system is a software application built upon planning and scheduling models and methods. It can be viewed as a particular case of information system (IS) supporting business functions [6, pp. 291–297].

In this chapter, we discuss APS systems. We list some of their general functionalities, and describe the conceptual architecture. Besides that, we introduce the typical forms the user interfaces of scheduling systems take.

4.1 Advanced Planning and Scheduling

The boundary between *planning* and *scheduling* is diffuse. Kreipl and Pinedo [31] explain terms “*medium term planning models*” and “*detailed scheduling models*” as follows:

A medium term production planning model typically optimizes several consecutive stages in a supply chain (i.e., a multi-echelon model), with each stage having one or more facilities. Such a model is designed to allocate the production of the different products to the various facilities in each time period, while taking into account inventory holding costs and transportation costs.

A short term detailed scheduling model is typically only concerned with a single facility, or, at most, with a single stage. Such a model usually takes more detailed information into account than a planning model.

The APS systems are software tools supporting planning and scheduling with optimization algorithms. While Hvolby and Steger-Jensen [3] state that no common definition of APS system exists, they name its main features:

APS systems utilize complex mathematical algorithms to forecast demand, to plan and schedule production within specified constraints, and to derive optimal sourcing and product-mix solutions. [...] Still, the decision-making is done by planners with insight in the particular supply chain and know how on the system constraints but likewise important: a feeling for feasibility of created plans.

The APS systems do not substitute other business IS that support planning and scheduling, such as Enterprise Resource Planning (ERP), but rather they supplement them. APS systems take over the planning and scheduling tasks, while ERP systems are still required as transaction and execution systems for orders [32]. Unlike the ERP system, the APS system typically has only a few users in the organization [33].

4.2 Functionalities

Framinan et al. [6, pp. 300–302] state that the main business functions targeted by the system (*scope of the system*) are based on the three-level classification of planning and scheduling introduced by Aytug et al. [34]: (i) production planning, (ii) release scheduling, and (iii) reactive scheduling. The first two levels predict the production plan and schedule, and the third is more involved with reacting to the current local situation (e.g., machines break down or managers submit high-priority items). The scheduling system should adequately cover all these levels, and functionalities regarding the monitoring and execution of the planned schedules should be integrated [6, pp. 300–302].

The general functionalities (or functional requirements) of scheduling systems were classified by Framinan et al. [6, pp. 300–315]. Selected functionalities are listed below.

Modeling requirements relate to the modelization of the shop floor. That is, the system should have the ability to work with a simplified subset of shop floor constraints; the system should transform schedules provided by the solution procedure into starting and finishing times of jobs.

Problem-solving functionalities are associated with the solution procedures. The system should support a group of algorithms for scheduling and rescheduling.

Solution-evaluation functionalities include the evaluation of solutions to different objectives and the analysis of scenarios (what-if analysis).

Capacity analysis functionalities refer to the ability to detect critical points and slacks in the schedule or instance. This includes the detection of potential bottlenecks and under-loaded sources.

Rescheduling functionalities include monitoring of the schedule execution and (automatic) triggering of rescheduling functions.

User interface requirements contain manipulation of schedules.

Integration with existing IS means that the system should be able to import the required data and export the results from (to) the other organizational systems, and it should perform the data checking.

The specific functionalities of the scheduling system depend on the industrial setting. The customer's needs may be satisfied with a customized generic system, or, in many instances, application-specific systems (or modules) have to be developed [35, pp. 476–479].

4.3 Architecture of a Scheduling System

According to Pinedo [35, pp. 460–467], a scheduling system typically consists of multiple different modules (see Figure 4.1). Various types of modules can be categorized as follows:

- database, object base, and knowledge-base modules,
- modules that generate the schedules, and
- user interface modules.

The scheduling tool is usually integrated with other organizational IS, such as shop floor control system, Material Resource Planning (MRP) system, or ERP system.

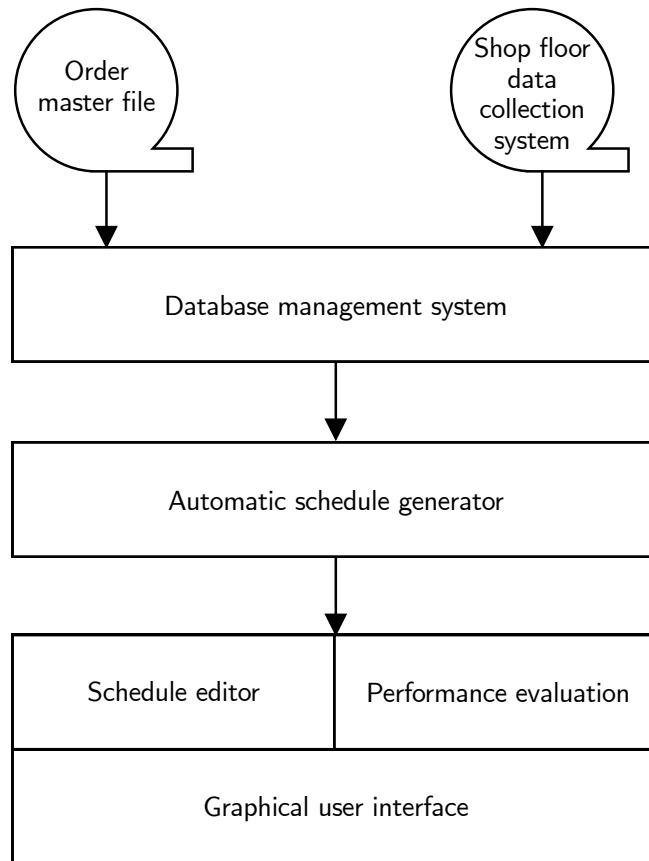


Figure 4.1: Configuration of a scheduling system. Adapted from Pinedo [35, p. 461].

Database management subsystem needs various basic functionalities, including multiple editing, sorting, and searching routines. Both data independent upon schedule (static) and dependent upon schedule (dynamic) are stored [35, pp. 462–467].

Modules that generate the schedules may utilize a variety of scheduling techniques, including hybrid approaches. One approach to scheduling is algorithmic or optimization, and the others are knowledge-based or constraint programming [35, p. 467].

User interface modules are further described in the following section.

4.4 User Interfaces of Scheduling Systems

According to Pinedo [35, p. 470], UI that present information regarding the schedules can take many different forms. Interactive scheduling tools typically use a *Gantt chart* as their primary interface for schedule manipulation [36]. Nevertheless, multiple interfaces are typically implemented in the APS system. For instance, Gantt chart, capacity buckets, and data table interfaces are implemented in Opcenter APS¹, a commercial APS system.

4.4.1 Gantt Chart Interface

Gantt charts (Figure 4.2) are typically presented as horizontal bar charts, with the horizontal axis representing the time and the vertical axis displaying the various resources. Users typically interact with the Gantt chart using a mouse. They can scroll over the timeline

¹<https://www.plm.automation.siemens.com/global/en/products/opcenter/aps.html>

or “drag and drop” activities within the schedule. However, manipulating solutions can have side effects, such as a cascading effect where moving one activity may require moving other activities as well to maintain feasibility [35, pp. 470–476].

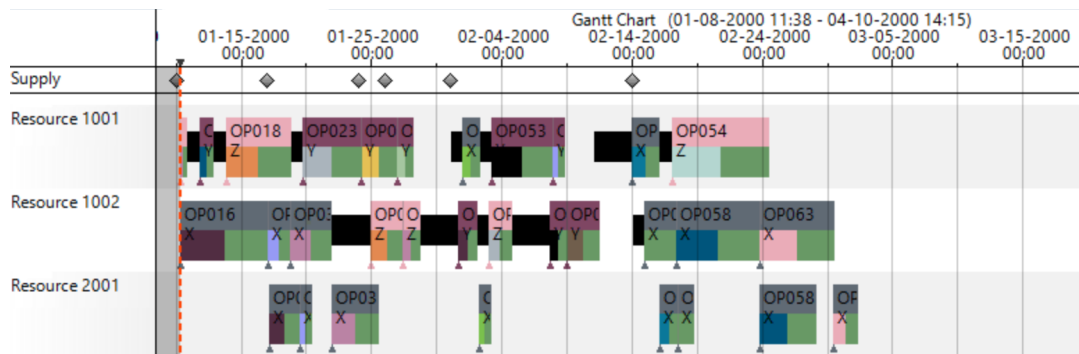


Figure 4.2: Gantt chart interface. Screenshot from Opcenter APS.

While Gantt charts are well-known and relatively easy-to-understand [6, pp. 323–324], they have limitations, especially in terms of scalability. In typical manufacturing schedules with dozens of resources and hundreds of activities on each resource, the screen can become cluttered and it may be difficult to recognize which bar corresponds to which job [7]. Furthermore, the capacity to display all parameters is rather limited, which may hinder the ability to make well-informed decisions [36].

4.4.2 Capacity Buckets Interface

In capacity buckets interface (Figure 4.3), the time axis is partitioned into a number of time slots or buckets. For each machine, the processing capacity of a bucket is known. The interface displays the percentage of the capacity utilized in each time segment [35, pp. 474–475]. It is primarily an output interface, and interactivity is not required [6, p. 325].

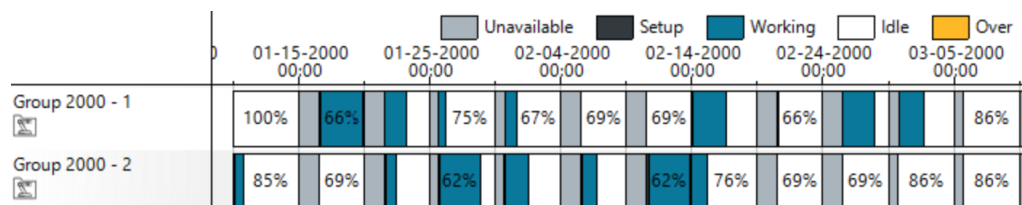


Figure 4.3: Capacity buckets interface. Screenshot from Opcenter APS.

4.4.3 Data Table Interfaces

The schedule can be displayed in a data table (Figure 4.4), i.e., a table where rows are orders or activities and columns are their parameters. Tables usually do not fit the screen, vertically nor horizontally. Thus, filtering, reordering, and hiding rows and columns and sorting rows should be easy to accomplish [37].

A dispatch list interface (Figure 4.5) is another form of displaying schedule information, similar to the data table. It is a list of the jobs to be processed on each machine in the order in which they are to be processed [35, pp. 472–474].

The data table interfaces do not have the disadvantages of Gantt charts since the user can find an activity in the list by its number. On the other hand, these interfaces do not

Order Status	Order No.	Part No.	Product	Due Date	Priority	Quantity	Op. No.	Operation Name	Operation Progress
Suggested	OP027	B2	SKU B2	Unspecified	10	210	10	Process Group 4000	Not Started
Suggested	OP028	C1	SKU C1	Unspecified	10	60	10	Process Group 4000	Not Started
Suggested	OP029	C3	SKU C3	Unspecified	10	120	10	Process Group 4000	Not Started
Suggested	OP030	X	SKU X	Unspecified	10	340	10	Process Group 1000	Not Started

Figure 4.4: Part of data table interface. Screenshot from Opcenter APS.

have the advantages of Gantt charts, i.e., the planner does not have a clear view of the schedule relative to the time [35, pp. 472–474].

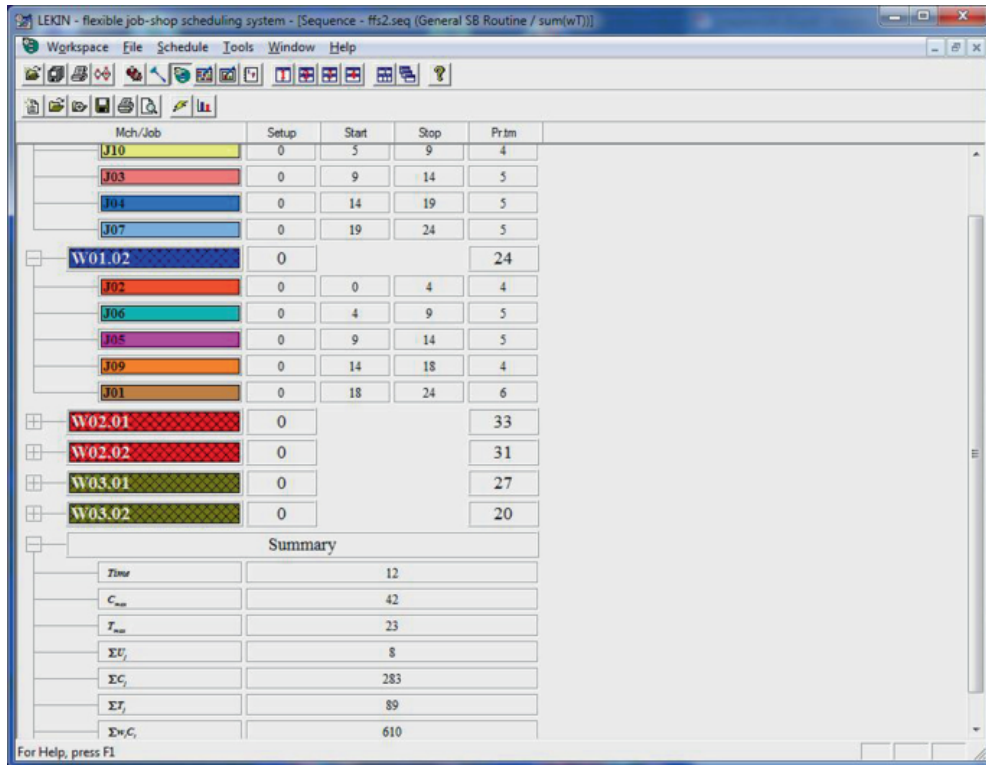


Figure 4.5: Dispatch list interface. From Pinedo [38].

4.5 User Interfaces in Commercial Systems

For reference, we also gathered several screenshots (Figures 4.6–4.10) from APS Fabio², a commercial APS system. It uses most of the basic components and views we enumerated in the previous section.

Views are organized using the *module tabs* pattern, only one module is visible at a time, and the user clicks on tabs to bring different modules on top [39, pp. 155–158]. The UI relies heavily on the use of data tables with conditional formatting, which is used to present the state of an operation (Figure 4.6, Figure 4.10) or to highlight a value (Figure 4.7, Figure 4.10). Cumulative data about the utilization of resources are presented in a capacity buckets interface (Figure 4.7).

The schedule is also presented as a Gantt chart (Figure 4.8, Figure 4.9). These screenshots support the belief that the capabilities of the Gantt chart to display the schedules

²<http://merica.cz/cs/products/fabrio/>

4 Planning and Scheduling Systems

in real-world environments are limited. For example, the visual representation appears cluttered, visual searching for a specific activity is difficult, or shorter activities are hard to be targeted with a pointer.

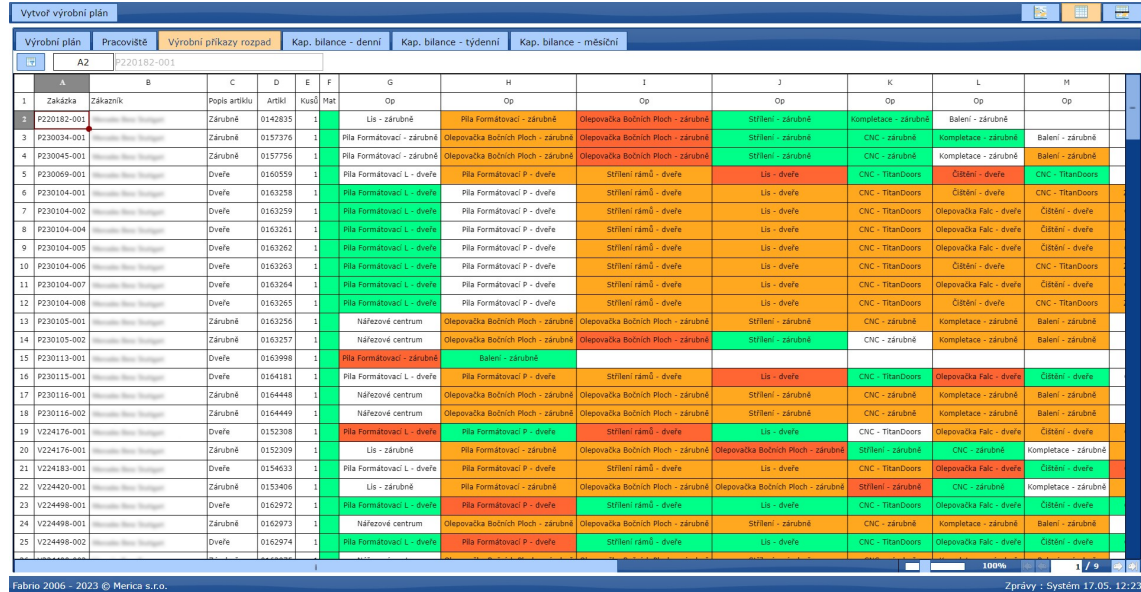


Figure 4.6: Overview of orders with a list of operations. Screenshot from APS Fabrio. Names of the clients were intentionally blurred.

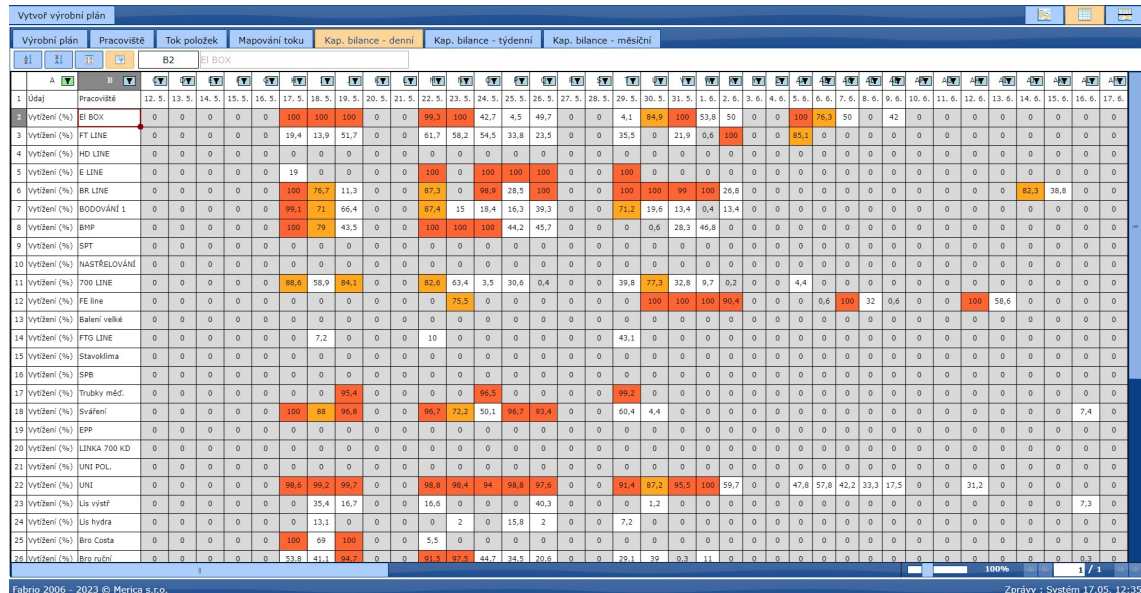


Figure 4.7: Capacity balance interface is using coloring for highlighting critical values. Screenshot from APS Fabrio.

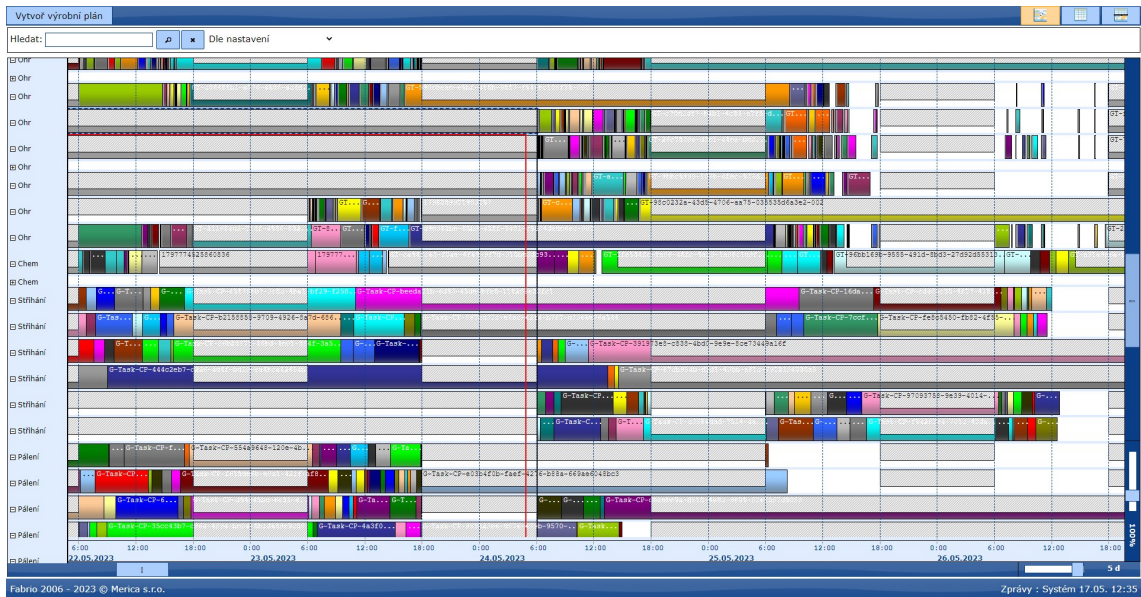


Figure 4.8: Gantt chart interface. Screenshot from APS Fabrio.

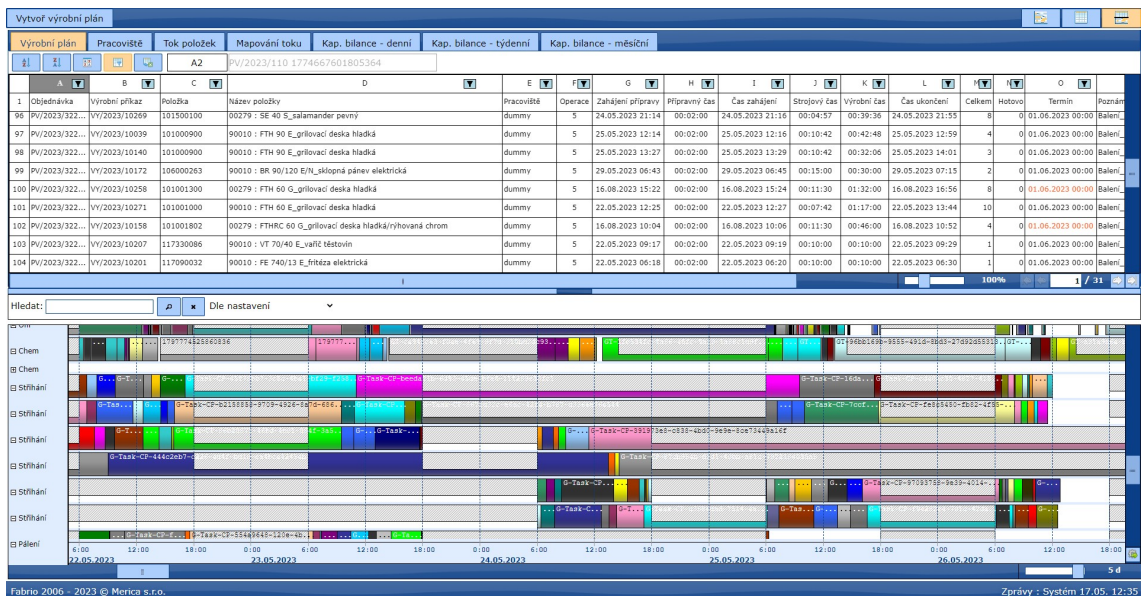


Figure 4.9: List of products and Gantt chart. Screenshot from APS Fabrio.

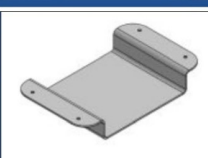
4 Planning and Scheduling Systems

Merica **Fabrio** Odraňovací lis Odhlásit se

Wyber jiné pracoviště

D24 Separation behind driver

	A	B	C	D	E	F	G	
1	Řada - příkaz	Stav operace	Číslo výrobku	Popis produktu	Umístění výrobku	Cil. množství	Real. množství	Vykázané
24	100*64424	Dokončena	5802365826	Separation behind driver		0	50	Ales B
25	100*64575	Uvolněna	BAC-0552-1000	PLECH PODVOZKU	A14, H23, H37	2		
26	100*64601	Uvolněna	BAC-0109-1000	příčka	H37, A31	28		
27	100*64032	Čekající	NS12 425 000002	držák 2 svorkovnic	A21, 302	2		
28	100*66004	Uvolněna	PCCT2_30w03	lichyt skluzu		1		Gustav
29	100*66528	Uvolněna	4L464-122	líta pro ptl		1		
30	100*62295	Čekající	MS-20180419-000	Vzpěra do lednice		65		
31	100*64721	Dokončena	11052914	Vana zásobník kovů 0300-500	A30, C07	0	20	Zdeněk
32	100*65527	Uvolněna	10680649	VÝZTUHA 3mm	A38	12		
33	100*63972	Chyba	5801776885	HOSE	A03	22		
34	100*66520	Uvolněna	4L463-122	Miska		1		
35	100*66517	Uvolněna	4L463-122	Držák piktalighu2		1		
36	100*66514	Uvolněna	4L463-117	Držák díla		1		
37	100*53783	Čekající	STP-00239_202	Papierhalter	A29, B06	272		
38	100*59714	Čekající	15206995_201	Retainer	A03	20	84	Gustav
39	100*62997	Čekající	NBN-399 435 0012-02-1	viko typ-1		1		
40	100*63115	Uvolněna	799-2803basketrib102	Basket rib	A25, N03	103	697	Gustav
41	100*58737	Chyba	50-2420-017_OD	LOADING BAY V4	A37, C17, 004	2		58 Zdeněk
42	100*64596	Uvolněna	BAC-0303-1000	věko	H37, A31	21		
43	100*65056	Uvolněna	10566874	Stěna boční levá	A41	3		
44	100*65058	Uvolněna	10566862	Stěna boční pravá	A41	3		
45	100*58738	Čekající	50-2420-017_OD	LOADING BAY V4	A37, C17, 004	60		



Fabrio 2006 - 2018 © Merica s.r.o. Zprávy - Systém 11.07. 21:27

Figure 4.10: List of operations on a resource. Screenshot from APS Fabrio.

5 Design Problem

In this chapter, we describe the findings we made throughout the design process with several UX design methods—personas (Section 5.2), scenarios (Section 5.3), task analysis (Section 5.4) and design requirements (Section 5.5).

5.1 Problem Statement

Marigold Engineering, Inc. is a well-established company in the machinery industry that processes large business-to-business projects and delivers highly-customized solutions to its customers. Given that their manufacturing processes are very complex, proper scheduling is necessary. Hence, they employ a professional planner who develops a production schedule on daily basis. While he uses an APS system in his job, the support for his tasks, such as improving a schedule, is inadequate, and his employer would like him to be more effective.

5.2 Persona Hypothesis

The characteristics and goals of expected end-users and relevant non-users are often represented as *personas*, fictive human beings. This user model is successful because it “*engages empathy of the design and development team around the users’ goals*” [40, p. 66]. While several personas may be identified, the interface design is mainly targeted at one *primary persona* whose needs and goals “*can be completely and happily satisfied by a single interface without disenfranchising any of the other personas*” [40, p. 87].

Traditionally, personas are constructed from qualitative research. Goals and other attributes are derived from the behaviors of users and potential users. According to Cooper et al. [40, p. 83], the most important distinction between behavior patterns of personas emerges by focusing on the following types of variables:

- *activities*—what the user does; frequency and volume,
- *attitudes*—how the user thinks about the product domain and technology,
- *aptitudes*—what education and training the user has; ability to learn,
- *motivations*—why the user is engaged in the product domain,
- *skills*—user abilities related to the product domain and technology.

The findings we made throughout the design process suggest that the design of the scheduling system should be targeted at the following personas:

- Rudolf, the production planner (primary persona),
- Milada, the manager (non-user persona).

We constructed the personas mainly from the domain experts’ knowledge of the users. Thus, these personas should be considered *provisional*—the demographics are not based on real data, and the goals and behaviors were obtained from domain expert interviews rather than users. Still, provisional personas yield better results than no user models [40, p. 97].

5.2.1 Primary persona: Rudolf, the production planner



Figure 5.1: Avatar of Rudolf¹.

Rudolf (Figure 5.1) is an employee of Marigold Engineering, Inc. in his mid-fifties. He has been working there as a production planner for nearly two decades.

Rudolf’s primary job responsibility is to develop a production schedule regularly. Therefore, the scheduling software helps him in his day-to-day job. Before his company adopted an APS system, he used to work with a spreadsheet application, and he still prefers to see the data in tabular form.

Moreover, he does not like rapid changes in the tool. Yet, after a bit of training, he is always able to learn to understand new patterns (e.g., what the orange frame means or that he should look at the second line of the table before doing anything else) and he eventually becomes more efficient.

In order to create a better schedule, he directly communicates with shop-floor operators very often (e.g., he can request increasing the production capacity).

Milada—just like anyone who is not familiar with the system—says “he does some magic he apparently understands and is somewhat successful in it”.

5.2.2 Non-user persona: Milada, the manager



Figure 5.2: Avatar of Milada¹.

Milada (Figure 5.2) is a manager in Marigold Engineering, Inc. in her early thirties. She has been working there since she was a business school student.

Milada’s main mission is to keep the clients satisfied, i.e., ensure they get their products in time. In general, she is responsible for fulfilling the business goals such as profit or retaining important customers. Nevertheless, she does not work directly with the APS system and is interested only in the results.

¹Image from <https://flaticon.com>.

5.3 Scenarios

Persona-based scenarios are narrative descriptions of one or more personas using the future product to achieve specific goals [40, pp. 105–106]. It is a specific story with developed characters—which have goals and motivations—context and a well-formed plot [41].

Scenario 1

Rudolf is about to develop a production schedule for December. Milada named him six orders that have to be finished before the factory is shut down for the Christmas holidays. The system’s initial solution did not meet this requirement, three of these orders would be finished after New Year’s Day.

He adds deadlines to the three tardy orders, clicks the button “Apply changes” and waits for nearly 20 minutes until he gets a new solution where all prioritized orders are finished in time but one. To free some capacity, he postpones the orders from a client who is, according to Milada, always late with payments. Now he gets a new schedule that is satisfactory.

Scenario 2

While Rudolf was bragging about a great schedule to his colleague, his phone started ringing. Milada was on the line. She told him that he has to rework it—a highly reputed client insists on an earlier delivery date. Rudolf goes to his computer, filters out this order, and tightens its deadline. He generates a new schedule, worse in some metrics. Nevertheless, both the client and Milada are happy.

Scenario 3

The order from client “Cars&Trucks” has to be finished as soon as possible. First, Rudolf filters out this order in the table and marks it as selected. He tries to increase its weight, but it does not help him. In the detailed table with individual jobs and their requirements, the production line PL-42 is displayed in red, meaning it is over-utilized. Hence, he looks at the capacity table and finds out that this production line operates only for 8 hours a day.

He calls the leader of PL-42 and asks him if it would be possible to increase the capacity somehow. The leader says his subordinates are willing to work on the first Saturday. Rudolf opens the calendar of resource PL-42, adds the first Saturday as a working day, and generates a new schedule. As he can see in the overview, the tardiness of this order is reduced.

5.4 Hierarchical Task Analysis (HTA)

According to Rosala [42], task analysis is “*a systematic study of how users complete tasks to achieve their goals*”. The information about goals and tasks is gathered by observing and interviewing users or domain experts. The analysis often results in a graphical representation called a task-analysis diagram. The way users accomplish their goals is captured through *plans*, which define what the order of steps is and which steps might be undertaken by whom. HTA is a particular type of task analysis where the tasks can be broken down into subtasks.

In the case of the manufacturing scheduling system, the user’s goal is to create a production schedule (Figure 5.3). The tasks the user completes are typical in interactive scheduling systems (see Subsection 2.2). The full version of HTA diagram is in Appendix A.

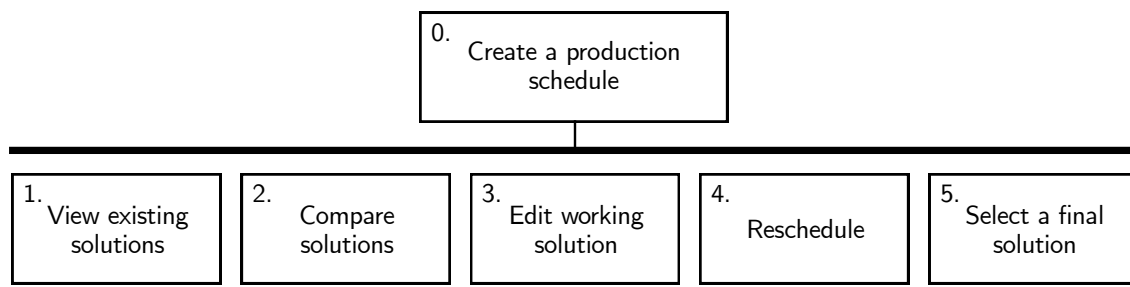


Figure 5.3: A short version of HTA diagram. Leaf nodes are underlined.

5.5 Design Requirements

Design requirements determine what information and capabilities personas require to accomplish their goals [40, pp. 106–107]. We extract the design requirements from *context scenarios*, which “describe the broad context in which that persona’s usage patterns are exhibited” [40, p. 113], including environmental and organizational considerations.

Design Requirements 1–4

Context scenario:

When Rudolf develops a new schedule, he uses some working solution as a basis. Unfortunately, the solution does not satisfy him completely, so he adds new constraints. While he knows some schedule improvement strategies, they are not always successful.

Implied needs:

- (i) Rudolf needs to have some working solution as a basis for the new problem model.
- (ii) Rudolf needs to change the constraints.
- (iii) Rudolf needs to run the rescheduling procedures at some time.
- (iv) Rudolf needs to return to previous solutions.

Design requirements:

- R1. User should develop a problem model based on the existing one.
- R2. User should be able to set some constraints manually.
- R3. User should be able to initiate the rescheduling at any time.
- R4. User should be able to return to previous solutions.

Design Requirements 5–6

Context scenario:

There are dozens of actions Rudolf can take to improve the proposed schedule. However, he is limited by time, and scheduling can be lengthy, so he has only a few attempts. He knows effective strategies for some cases, but he has to identify the situation. Sometimes, the problem is in capacity, and sometimes there is a lack of material, etc.

Implied needs:

- (i) Rudolf needs to be effectively supported in the decision-making process.
- (ii) Rudolf needs to identify the situation.
- (iii) Rudolf needs to see several representations of the schedule.

Design requirements:

- R5. User should be supported in the decision-making process.
- R6. User should be given multiple views on the schedule.

Design Requirement 7

Context scenario:

Scheduling can be a lengthy process, and the optimization procedure can take up to twenty minutes. During that twenty minutes, Rudolf might focus on something else. Hence, he sometimes forgets what parameters he has changed.

Implied need:

- (i) Rudolf needs to see what parameters of the problem he has changed.

Design requirement:

- R7. User should see what changes he has done.

Design Requirements 8–10

Context scenario:

When Rudolf develops a schedule, he considers a few alternatives. He has to pick one final solution, yet his time is limited. In addition, some clients are more profitable than others, and Rudolf has to make sure they are satisfied.

Implied needs:

- (i) Rudolf needs a method to evaluate a solution quickly.
- (ii) Rudolf needs to compare multiple alternative solutions.
- (iii) Rudolf needs to see if the orders from particular clients are processed optimally.

Design requirements:

- R8. User should be able to assess the quality of a solution.
- R9. User should be able to compare metrics of multiple solutions.
- R10. User should be able to assess the quality of a solution for particular orders.

Part II

Prototype

6 Model of Scheduling Problem

Proper validation of design ideas would not be possible without an appropriate scheduling problem to be solved. Therefore, we modified Resource-Constrained Project Scheduling Problem (RCPSP) to serve as a model for a real-life industrial scheduling problem.

6.1 Resource Constrained Project Scheduling Problem

RCPSP is formally defined by a 6-tuple $(V, p, E, \mathcal{R}, B, b)$ [43]. A visualization of the RCPSP definition is presented in the following figure (some of the edges are omitted as precedence relations are transitive):

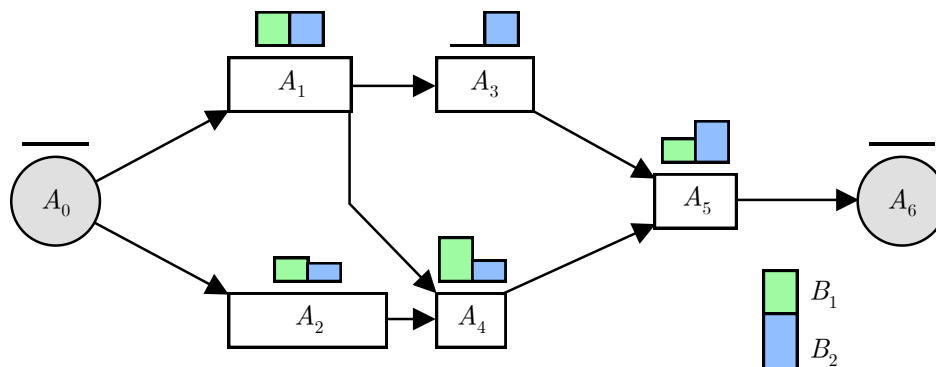


Figure 6.1: Visualization of RCPSP definition. Width of a node (activity) corresponds to its processing time; resource consumption is above each node; edges represent precedences. Adapted from Laborie [44].

Activities of the project are defined by a set $V = \{A_0, A_1, \dots, A_n, A_{n+1}\}$ where activities A_0 and A_{n+1} are dummy activities representing the start and the end of the schedule, respectively.

The set of non-dummy activities is identified as $\mathcal{A} = \{A_1, \dots, A_n\}$.

Durations are represented by a vector $p \in \mathbb{N}^{n+2}$, where p_i is the duration of the activity A_i and $p_0 = p_{n+1} = 0$ by definition.

Precedence relations are given by a set E such that a tuple $(A_i, A_j) \in E$ if and only if A_i precedes A_j . The precedence relations can also be represented by a directed graph $G = (V', E')$ where vertex $v \in V'$ corresponds to an activity $A_v \in V$ and arc $e = (i, j) \in E'$ corresponds to a precedence relation $(A_i, A_j) \in E$.

The activities A_0 and A_{n+1} are predecessors and successors to all, respectively.

Renewable resources are represented by a set $\mathcal{R} = \{R_1, \dots, R_q\}$.

Availabilities of resources are represented by a vector $B \in \mathbb{N}^q$ such that B^k denotes availability of R_k .

Demands of activities for resources are formalized by b , a $(n+2) \times q$ integer matrix, b_{ik} represents the amount of resource R_k used per time unit during the execution of A_i .

Schedule is a vector $S \in \mathbb{R}^{n+2}$ such that S_i represents the start time of activity A_i . $C_i = S_i + p_i$ denotes the completion time of activity A_i .

A feasible solution of RCPSP has to be compatible with both the precedence constraints (Equation (6.2)) and the resource constraints (Equation (6.3)) [43]. Equation (6.1) represents the set of non-dummy activities in process at time t .

$$\mathcal{A}_t = \{A_i \in \mathcal{A} \mid S_i \leq t < S_i + p_i\} \quad (6.1)$$

$$S_j - S_i \geq p_i, \quad \forall (A_i, A_j) \in E, \quad (6.2)$$

$$\sum_{A_i \in \mathcal{A}_t} b_{ik} \leq B_k, \quad \forall R_k \in \mathcal{R}, \forall t \geq 0. \quad (6.3)$$

The objective of RCPSP is to find a non-preemptive (activities cannot be paused during their execution) schedule S of minimal makespan $S_{n+1} = C_{n+1} = C_{\max}$ subject to precedence constraints and resource constraints [43]. In the 3-field problem classification introduced by Graham et al. [45], RCPSP can be viewed as $PS \mid \text{prec} \mid C_{\max}$.

In the case of the instance defined by Figure 6.1, a visualization of the optimal solution is provided in Figure 6.2.

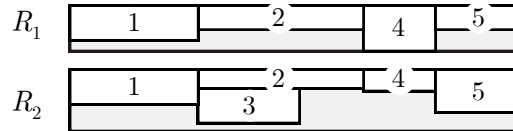


Figure 6.2: Visualization of RCPSP solution for the instance defined in Figure 6.1.

The Project Scheduling Problem Library (PSPLib)¹ provided initial problem instances for RCPSP.

6.2 Problem Modifications

While the standard RCPSP is one of the most intractable classical scheduling problems [43], it does not simulate the real-life industrial scheduling problem users solve in APS sufficiently. We aim to model a production plant where orders from multiple clients are processed, having some shipping dates and priorities. The factory has multiple production lines where employees work in varied shifts. Production is continuous, so the C_{\max} criterion is insignificant.

Thus, we altered the RCPSP both in the objective, which is dependent on weight w_j and due date d_j , and hard constraints (resource calendars, deadlines \tilde{d}_j , release times r_j). In addition, the precedence graph is split into multiple components representing orders from different clients. The parameters d_j , w_j , \tilde{d}_j , and r_j are assigned to components instead of individual activities.

In summary, the process of instance modification consists of multiple tasks:

- relaxing some precedence relations,
- adding due dates to components,
- adding weights to components,
- adding operating hours to resources, and
- adding deadlines and release dates to components.

¹<https://www.om-db.wi.tum.de/psplib/main.html>

Precedence graphs of RCPSP instances are connected, i.e., they have a single component. However, to better model the existence of several orders consisting of multiple operations, some of the precedence relations are removed, and the graph is split into a *set of components* $\mathcal{M} = \{M_1, \dots, M_r\}$, where $M_j \subseteq \{0, 1, \dots, n+1\}$ and $M_j \cap M_l = \emptyset$ if $j \neq l$. Multiple PSPLib instances can be merged into one to create a more complex instance.

Each of the graph components has a due date assigned. The generation procedure uses a scheme proposed by Hall and Posner [46]. In this scheme, the due date d_j of the component M_j depends on a constant $K \geq 0$, a functional h_j and a random variable X_j (Equation (6.4)).

$$d_j = \begin{cases} K & \text{if } j = 1, \\ h_j(p_1, \dots, p_n, d_{j-1}) + X_j & \text{if } j > 1. \end{cases} \quad (6.4)$$

In our implementation, the components are sorted randomly, h_j depends on the length of the critical path of the component M_j (i.e., the longest path between the first and the last node in the precedence graph), and X_j is normally distributed ($X_j \sim \mathcal{N}(-4, 10)$ by default, but parameters can be changed).

Resources can process activities only at given times. Due to that, tasks have to be pre-emptive—execution of activity A_i is paused when any of the resources demanded by A_i becomes unavailable and continues once all necessary resources are in operation. According to Kreter et al. [47], the calendar for resource $R_k \in \mathcal{R}$ is a step function $\mathbf{Cal}_k(\cdot) : [0, \infty) \rightarrow \{0, 1\}$ continuous from the right at the jump points, where the condition

$$\mathbf{Cal}_k(t) = \begin{cases} 1, & \text{if period } [[t], [t+1]) \text{ is a working period for } R_k, \\ 0, & \text{if period } [[t], [t+1]) \text{ is a break period for } R_k, \end{cases} \quad (6.5)$$

is satisfied. An activity calendar $\mathbf{Cal}_i^V(\cdot) : [0, \infty) \rightarrow \{0, 1\}$ is derived from the resource calendars as follows:

$$\mathcal{R}_i = \{R_k \in \mathcal{R} \mid b_{ik} > 0\}, \quad (6.6)$$

$$\mathbf{Cal}_i^V(t) = \begin{cases} \min_{R_k \in \mathcal{R}_i} \mathbf{Cal}_k(t), & \text{if } \mathcal{R}_i \neq \emptyset, \\ 1, & \text{otherwise.} \end{cases} \quad (6.7)$$

6.3 Constraint Programming Model

Constraint satisfaction problem \mathcal{P} is a triple $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where

- $\mathcal{X} = (x'_1, \dots, x'_n)$ is a n -tuple of variables,
- $\mathcal{D} = (D'_1, \dots, D'_n)$ is a n -tuple of domains such that $x_i \in D'_i$, and
- $\mathcal{C} = (C'_1, \dots, C'_t)$ is a t -tuple of constraints.

A constraint $C'_i = (R'_i, S'_i)$ is a relation R'_i defined on a scope S'_i , $S'_i \subseteq \mathcal{X}$ [48, p. 25].

In the modified version of RCPSP, the variables are starting times of activities, and the domains are $D'_i = \{r_i, r_i + 1, \dots, \min(\tilde{d}_i, \text{UB})\}$. In this model, tardiness (Equation (6.8)) depends on the completion time of the component; the function $C_i(S_i)$ (Equation (6.9)) is the completion time of activity A_i if the execution starts at time S_i .

$$T_j = \max_{i \in M_j} (T_i) = \max_{i \in M_j} (C_i - d_j, 0), \quad (6.8)$$

$$C_i(S_i) = \min \left\{ t \mid \sum_{\tau=S_i}^{t-1} \mathbf{Cal}_i^V(\tau) = p_i \right\}. \quad (6.9)$$

6 Model of Scheduling Problem

The objective is to minimize the weighted tardiness of components (Equation (6.10))

$$\min \sum_{j=1}^{|\mathcal{M}|} w_j \cdot T_j, \quad (6.10)$$

subject to the precedence constraints (Equation (6.11)), the calendar constraints (Equation (6.12)), the resource constraints (Equation (6.13)) and the constraints given by deadlines (Equation (6.14)) and release times (Equation (6.15)).

$$S_j \geq C_i(S_i), \quad \forall (A_i, A_j) \in E, \quad (6.11)$$

$$\sum_{t=S_i}^{C_i(S_i)} \mathbf{Cal}_i^V(t) = p_i, \quad \forall A_i \in V, \quad (6.12)$$

$$\sum_{A_i \in \mathcal{A}_t} b_{ik} \leq B_k, \quad \forall R_k \in \mathcal{R}, \forall t \geq 0, \quad (6.13)$$

$$\max_{i \in M_j} (C_i) \leq \tilde{d}_j, \quad \forall M_j \in \mathcal{M}, \quad (6.14)$$

$$\min_{i \in M_j} (S_i) \geq r_j, \quad \forall M_j \in \mathcal{M}. \quad (6.15)$$

7 Solution Prototype I

This chapter describes the first version of the interactive scheduling system prototype. The purpose of this prototype was to model a simple UI of a scheduling system that allows the users to interactively improve the problem model. Prototype evaluation is described in Chapter 8.

7.1 Key Concepts

We expect the user to select some working solution, edit model parameters, and trigger scheduling procedures—and repeat this until he finds a solution that satisfies him.

The user can select any existing solution and build a new model on its basis. After he finishes building a model, he can trigger the solver and generate a new version of the solution. Hence, there is a relation between different problem models; a model has all parameters of its predecessor and some additional changes. Tracking changes and the ability to recall any previous model is the basic idea of version control systems (VCS) such as Git¹, which inspired us.

Interactions with a schedule are indirect. We do not allow for manipulating the output for several reasons:

- moving one activity in time may cause cascading effect,
- execution of orders as a whole is more important than the execution of activities,
- moving one activity may not help with the execution of the whole order,
- moving the whole order means moving multiple activities, which is inconvenient.

Instead, the user can steer the optimization procedure by setting several constraints of the problem (*release times*, *deadlines*, *order weights*, and *resource calendars*).

Deadlines and release times help the user with moving the orders in time. Deadlines are useful when he wants to ensure that the order gets finished in time or no schedule is generated. With release times, the user can intentionally delay the execution of some orders, for example, when problems with capacity occur. When the user sets both the release time and deadline to the order, he locks the order into a specific time interval.

Order weights can directly adjust the tardiness penalization in the objective function (see Equation (6.10)). It is expressed by number for testing purposes, although typically, it is not necessary to have more than three categories [35, p. 462].

Resource calendars allow for changing the operation hours of resources. Initially, there are two types of resources modeled—the ones with one shift only (8 am to 4 pm) and the ones with first (6 am to 2 pm) and second (2 pm to 10 pm) shifts, and the production plant is not in operation on weekends. However, the user can manipulate this to create a better schedule, and he can add overtime hours, temporarily add a second shift to resources with one shift, etc.

¹<https://git-scm.com/>

7.2 Prototype Technologies

The coded prototype is a web-based Python application built upon the architecture of a scheduling system (see Section 4.3). The pain point of APS systems we attempt to solve is not in their general functionality but in presenting information about schedules. Thus, the main objective of prototyping was to test the UI design ideas in an interactive system that simulates the scheduling capabilities of an APS system rather than creating functional software. We chose technologies that make simulating scheduling capabilities and building user interfaces easy. We consider Python suitable for rapid prototyping as well as manipulating large amounts of data.

Our representation of the database module is very simple, because storing data is not the subject of this work. We store modified PSPLib instances (see Section 6.2), lists of changes, and solutions as text files with prescribed serialization format.

We utilize the application programming interface (API) of IBM Decision Optimization CPLEX Modeling for Python (or DOcplex for short)², which implements modeling features specialized to scheduling, and for standard problems, it finds solutions that are comparable to solutions found by state-of-the-art specialized algorithms [49]. We use its no-cost academic edition for simulating the scheduling capabilities of the APS system and generating solutions for the constraint programming model described in Section 6.3. Source codes are part of Appendix C.

For building the user interface, we use Dash framework³, a low-code framework for rapidly building data apps in Python [50], with open-source libraries. We use only the free plan, and thus, the application cannot be deployed to production [51]. However, the debug environment was sufficient for our use case.

7.3 UI Components

This section introduces the UI components of our scheduling system. An overview of the information architecture is given in Figure 7.1. We attempted to support the decision-making process by providing the most important information on a single page (Figure 7.2), so the context was visible; the interfaces we considered less important are on separate pages.

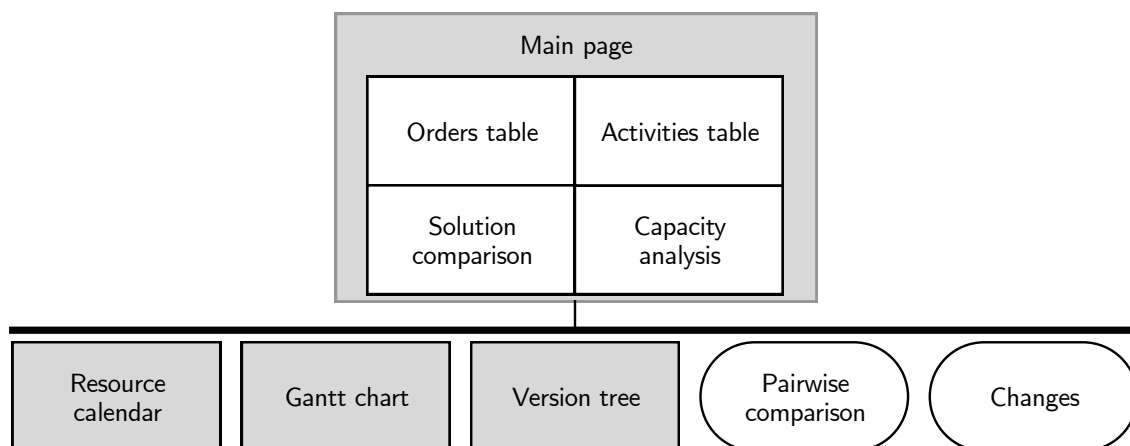


Figure 7.1: Scheme of information architecture. Gray rectangles are separate pages; rounded-corners-rectangles represent modal windows.

²<https://www.ibm.com/docs/en/icos/12.9.0?topic=docplex-python-modeling-api>

³<https://dash.plotly.com/>

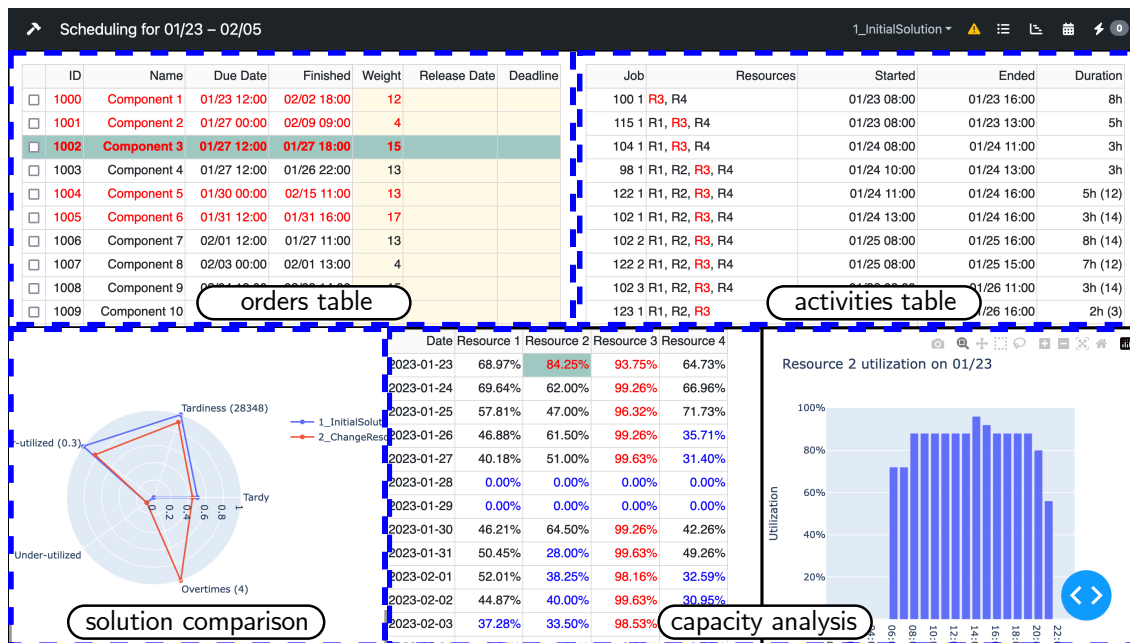


Figure 7.2: The main page screenshot with labels.

7.3.1 Orders Table and Activities Table

The orders table and activities table are data table interfaces, displaying information about the execution of orders and activities, respectively.

The tables are interconnected with *two-panel selector* pattern—when the user selects any order, the activities from this order are immediately shown on the second panel. This allows the users to quickly shift their attention between the overview of orders and the details about the execution of one particular order. The selected order stands out, having a different color than the rest of the table. Switching between orders can be accomplished with a single mouse click or by keyboard using the arrow key [39, pp. 198–201].

Both tables utilize conditional formatting. Delayed orders are highlighted in red in the orders table, making them visually salient. In the activities table, the resources that may potentially cause bottlenecks are colored in red.

The orders table allows for inline editing of several parameters (weight, release date, and deadline); the editable fields are highlighted in yellow. The text fields use the *forgiving format* pattern [39, p. 355], permitting a variety of date and time formats and shortcuts (e.g., if today is April 28, 2023, the date May 19, 2023, 12:00 am can be inputted either as 2023-05-19, 23/05/19, 05/19 00:00, etc., or with shortcuts such as 19 and 19 0).

Orders can be marked as prioritized in the table; this links them to the solution comparison interface—if the user picks a subset of orders, he can compare their cumulative tardiness between solutions.

There may be multiple rows for one activity in the activities table—it depicts that the execution was paused because some resource became unavailable and continued later.

7.3.2 Solution Comparison

The solution comparison interface provides a big-picture view of multiple solutions, allowing the user to compare them in multiple key performance indicators (KPI) (solution tardiness, percentage of tardy orders, number of overtime hours, percentage of over-utilized and under-utilized resources, priority order tardiness, tardy priority orders).

7 Solution Prototype I

The data are displayed in a *polar chart*—each variable is provided with an axis that starts from the center, and the axes are arranged radially, with equal distances between each other [52]. Each solution is represented by a polygon; solutions in the center are the most desirable, having optimal utilization of resources, low number of overtime hours, and low tardiness. Theoretically, the number of solutions that can be compared in a single chart is not limited. However, the user can hide some solutions from the visualization by unselecting them in the legend or the version tree interface.

7.3.3 Capacity Analysis

Capacity analysis interface is an example of *capacity buckets interface* (see Subsection 4.4.2), where the vertical axis, representing time, is divided into days, and the various resources are displayed on the horizontal axis. For each day and resource, the production capacity usage for the entire shift is presented as a percentage with two decimal places. Data are presented in a table with conditional formatting, over-utilized and under-utilized resources are highlighted in red and blue, respectively.

If the user clicks on any cell, the cell gets highlighted, and the utilization on the given day and resource is shown in more detail in a *bar chart* divided to 24 hours on the *x-axis*, allowing to examine the utilization on every hour of the day.

7.3.4 Pairwise Comparison

Pairwise comparison (Figure 7.3) is a *modal window* allowing to compare differences between two problem models. This functionality was inspired by VCS. In VCS, it is possible to compare changes between source code versions anywhere in the version tree. In this prototype, this feature is less sophisticated; differences between any two versions of the problem model are shown in two columns; the first column shows changes that are present in the first version and not in the second one and vice versa. The solutions for the problem model are compared in a polar chart (similar to the polar chart described in Subsection 7.3.2).

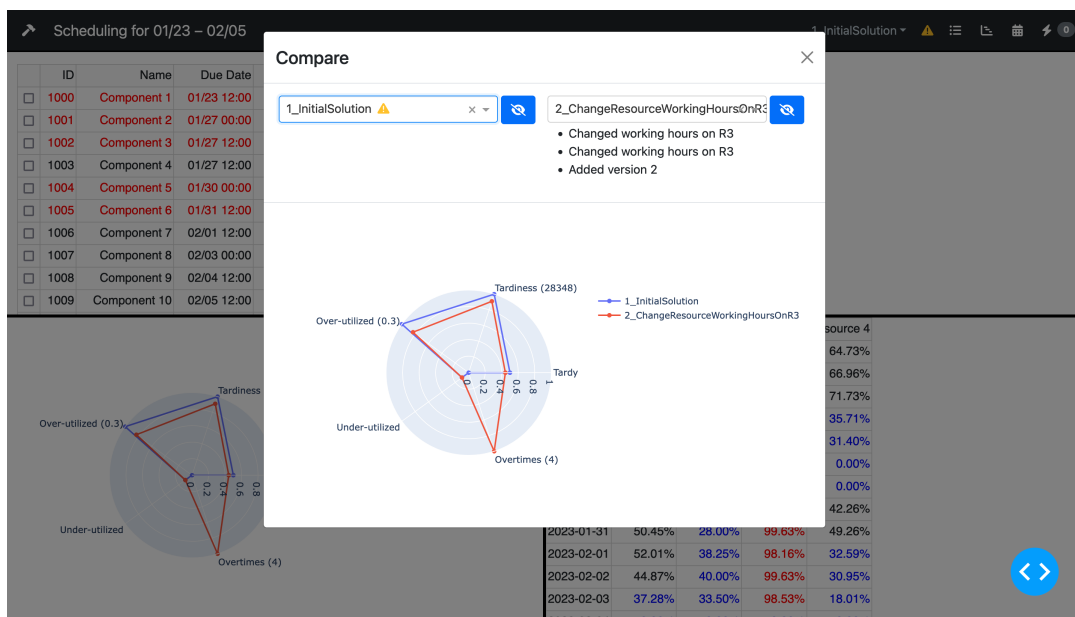


Figure 7.3: Pairwise comparison window.

7.3.5 Changes

Changes interface (Figure 7.4) is a *collapsible panel*, which contains a list of changes the user performed on the selected solution version and two buttons. The “Discard” button removes all the changes, while the “Apply changes” button triggers the scheduling procedure and creates a new solution version.

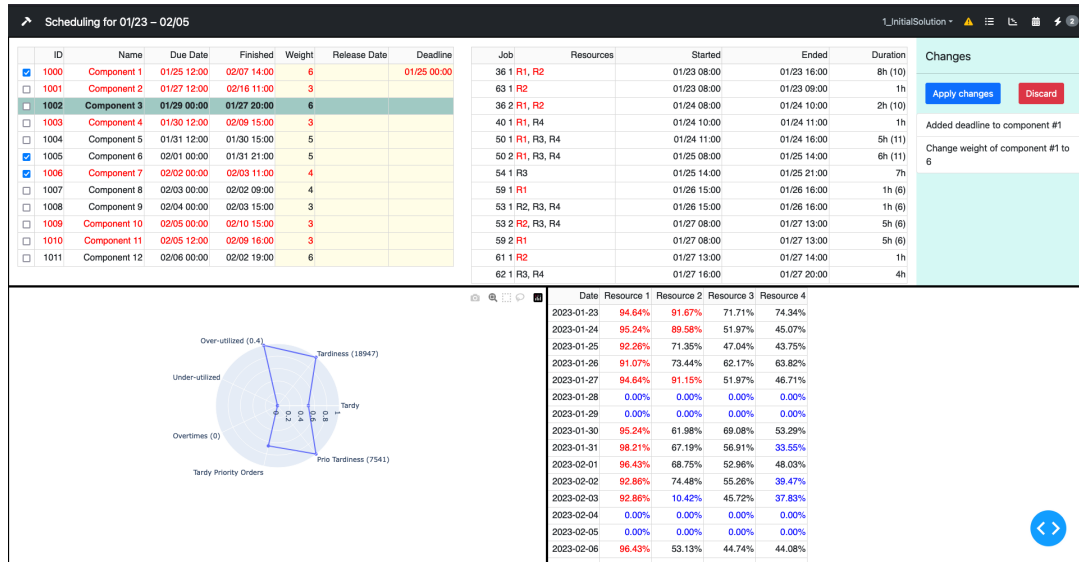


Figure 7.4: Main page with changes panel open.

7.3.6 Resource Calendar

The resource calendar is a table with dates and start and end times, allowing one to view and edit the operation days and hours of resources. The user can switch between resources using the *select* component on the top.

Resource 1														
	<input checked="" type="checkbox"/> 05/01	<input checked="" type="checkbox"/> 05/02	<input checked="" type="checkbox"/> 05/03	<input checked="" type="checkbox"/> 05/04	<input checked="" type="checkbox"/> 05/05	<input type="checkbox"/> 05/06	<input type="checkbox"/> 05/07	<input checked="" type="checkbox"/> 05/08	<input checked="" type="checkbox"/> 05/09	<input checked="" type="checkbox"/> 05/10	<input checked="" type="checkbox"/> 05/11	<input checked="" type="checkbox"/> 05/12	<input type="checkbox"/> 05/13	<input type="checkbox"/> 05/14
From	8	8	8	8	8			8	8	8	8	8		
To	16	16	16	16	16			16	16	16	16	16		

Figure 7.5: Resource calendar.

7.3.7 Gantt Chart Visualization

The solution is visualized as a Gantt chart (Figure 7.6). The activities are represented as lines, and their attributes are encoded visually as follows:

- the position on the x -axis gives the start time,
- the length of the line represents the processing time,
- the thickness of the line shows the amount of resource used for the execution of the activity,
- the color represents the tardiness of the order.

The visualization utilizes the following interaction patterns:

Overview plus detail—an overview of the Gantt chart is always shown at the bottom.

The user can display a part he is interested in in the detail view [39, p. 296].

7 Solution Prototype I

Data spotlight—activities from one order are highlighted when the user hovers his mouse over any part of the order, while activities from other orders are dimmed [39, p. 303].

Datatips—when the user hovers his mouse over any activity, data values for that activity are displayed in a tooltip [39, p. 299].



Figure 7.6: Gantt chart visualization.

While the activities can usually be modeled as non-overlapping rectangles, it is not guaranteed that such visualization exists. The RCPSP does not treat the activities as geometric rectangles because activities are not necessarily assigned to the same resource units over their processing times [53]. Csébfalvi [54] presented a theoretically correct method for resource utilization visualization; the method suggests dividing activities into strips and solving the strip packing of strips problem. However, that would add more undesirable complexity to the visualization. Thus, the rectangles representing the activities can overlap on the Gantt chart in such edge cases.

Unlike the standard Gantt chart interface described in Subsection 4.4.1, this interface is read-only, i.e., it does not allow any modifications to the solution.

7.3.8 Version Tree

Version tree is another idea adopted from VCS. The user selects any working solution (equivalent to “checkout” in VCS) and adapts it. Then, the new version has all parameters of the original problem model and new parameters given by the changes. This relation is represented as a *multilevel list* (Figure 7.7).

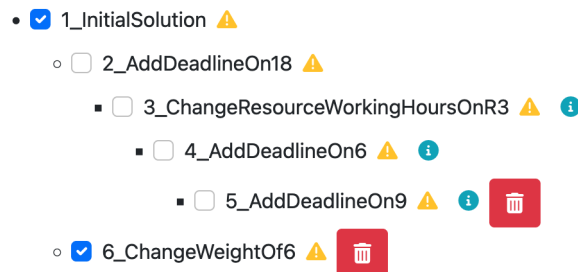


Figure 7.7: Version tree.

The versions are selectable; the user can select versions he wants to compare in the main interface. He can also delete the leaves of the tree if he considers the solution insufficient.

8 Prototype I Evaluation

In this chapter, we discuss the testing of the first version of the solution prototype, which we described in Chapter 7. Additional documentation, containing the document with instructions we sent to the users for the asynchronous study (in Czech) and a summary of the testing, is a part of Appendix B.

8.1 Methodology

Prior to testing with users, we formulated the goals of the evaluation. Namely, we wanted to find out

- (i) whether they can create a schedule matching the given criteria in this interface,
- (ii) what questions about the data do they have,
- (iii) what opportunities for the improvement of visualization components are there, and
- (iv) what interface flaws cause problems.

Furthermore, we wanted to collect as much insight into the problematics of interaction in scheduling systems as possible. Thus, our goals were mainly exploratory, and we wanted to find directions for future iterations of design rather than validate the current interface.

Before the testing, we prepared three testing problem instances (see Chapter 6). We formulated the goals for scenarios—e.g., we determined a subset of orders that should be finished in time. We introduced the interface to the participants and gave them instructions for testing—we asked them to fulfill the scenarios in a simple user interface prototype.

We recruited only the users ($N = 2$) who are domain experts in scheduling and have already been familiar with an interface of APS system, i.e., novices did not participate in the testing. Both users were involved in the previous phases of design. Their profiles are in Table 8.1; we asked them about their age, current occupation, and years of professional experience with scheduling for industry.

	Sex	Age	Current occupation	Experience
P1	M	56	manager	10 years
P2	M	51	sales manager, analyst	21 years

Table 8.1: Profiles of first coded prototype testers.

The prototype was tested asynchronously; the users were requested to write self-reports, i.e., what they did and what problems they encountered. We chose the asynchronous testing over other methods because it has the benefit of flexibility—the users can test on their own. Moreover, we conducted a group interview with the users afterward.

8.2 Results

We let the users try the system independently and collected feedback from their self-reports and the group interview.

Are the users able to create a schedule matching the given criteria in this interface?

The users reported that they created schedules matching the criteria we requested. Thus, they presumably finished the scenarios successfully.

What questions about the data do they have?

The users had several questions about the schedule, such as:

- How tardy will a given order be?
- What will the slack of a given activity be?
- How will an activity contribute to the overall tardiness of an order?
- When will the overtime hours be added?
- How many orders will be ready on a given resource by the start of the day?
- What proportion of tardy orders will be processed on a given day and resource?
- How will the resource be utilized every hour?

What opportunities for the improvement of visualization components are there?

The users reported that the interface performs as they expected—we included the ideas we previously discussed with them. However, there appear to be several opportunities for future iterations of the design, with improving the *capacity analysis interface* being the most notable.

The *orders and activities tables* (Subsection 7.3.1) appear to be an appropriate representation of the orders and the activities. The users did not report any particular problem with using them. Nevertheless, we uncovered additional information that could be shown in the tables, such as the numeric value of tardiness for both orders and activities, and the slack of an activity.

The *solution comparison* (Subsection 7.3.2) is represented as a polar chart, which, in the current implementation, has several limitations. The users reported that they do not understand the values and they would require some additional explanation on what the maximum value means. They also suggested having a table with the numeric values of KPIs in an additional table, as these are not presented in the graph.

Testing showed a limitation in our implementation of the *capacity analysis interface* (Subsection 7.3.3). The users reported that they used it to search for over-utilized resources. This interface provides them with information about the utilization of a resource in one day. This information is expressed as a percentage with two decimal places and supported by highlighting the value if the resource is over-utilized or under-utilized. However, the users did not require such precision. Instead, they had other questions regarding the execution of orders on a given resource, such as the number of orders ready by the start of the day or the proportion of tardy orders processed on a given day. They also stated that the 24-hour utilization graph is too big for the information it gives. In conclusion, there is an opportunity to use the space available more effectively and display more information.

We also uncovered a possibility to extend the functionality of *pairwise comparison* (Subsection 7.3.4) by implementing the option to merge two versions of the problem model into one.

The *resource calendar* (Subsection 7.3.6) could be a part of the main interface instead of being a separate interface. It could also be used for highlighting when the overtime hours have been added.

There are also several opportunities for improving the *Gantt chart* (Subsection 7.3.7). One user reported that he used it to find the critical path; thus, we could express the precedences and the critical path in this interface. We could also implement advanced filtering features.

A simple multilevel list with bullets seems to be an inappropriate representation of a *version tree* (Subsection 7.3.8). One participant used this interface, and yet he asked us where the tree was, and he apparently did not perceive it as a tree but rather as a list. Thus, we should highlight the structure of the tree in a better way. Additionally, it should not be a separate interface. Instead, it could be a part of the main interface, more interconnected with the solution comparison component.

What interface flaws cause problems?

We did not uncover any usability catastrophes or any major issues, but we detected several usability problems that require further attention:

- (i) insufficient representation of overtime hours (minor), and
- (ii) variable number of axes in the polar graph (minor).

Participants reported that the *representation of overtime hours is insufficient*. The information architecture intensifies this flaw, as the resource calendar is a separate window, and thus, nothing in the main interface reminds the users that they should try to reduce the number of overtime hours. Therefore, we should highlight the extra working hours in the main interface.

The users were surprised that *the number of axes in the polar graph is variable*. This problem occurs when they select the first priority order (two additional axes are added) or unselect the last one (the two are removed). The shift in positions of the other axes could limit the users' ability to learn how to read the graph effectively. Therefore, we suggest keeping the number of axes constant, regardless of the number of selected priority orders.

8.3 Discussion

We collected feedback from users with a background in production scheduling. While they could finish the tasks in this interface, they discovered multiple opportunities for improvement, the most remarkable one being the change of information architecture.

The findings seem to be relevant, but there are two possible sources of bias—one is given by the nature of the methods, the other comes from the concrete test execution.

According to Bruun et al. [55], fewer usability issues are discovered with asynchronous usability testing than with conventional laboratory testing. Furthermore, self-reporting is susceptible to *self-reporting bias*. Some people—even if they are willing to help—do not feel comfortable admitting failure or revealing that they do not know something. However, the effects of self-reporting bias can be mitigated—one way is to follow up with additional interviews [26, pp. 256–257]. Thus, we compensated for the disadvantage of the method with a group interview, which allowed us to discover more problems.

The recruitment gives one potential source of bias originating in the concrete test execution. Since we recruited only two evaluators, who were moreover involved in the previous phases of design, the results possibly represent only the needs of a small group of potential users. In other words, the external validity of this study, i.e., “*the extent to which the results can be generalized to other populations and settings*” [56, p. 89], is limited.

Another potential source of bias is that the tasks were relatively straightforward compared to the reality the users typically face. For example, we assumed that the Gantt chart is a not very helpful component, but one of the evaluators reported that he used it to find the critical path. This finding can either mean that the problem instance was so simple that the Gantt chart was clear enough, so he could find the information he needed there, or that our assumption was incorrect and the Gantt chart is actually useful.

9 Solution Prototype II

Based on the results of the first evaluation of the prototype, we attempted to redesign the interface. We focused on the statistics provided to users and the overall information architecture. We have not considered some user ideas, such as improvements to the Gantt chart or merging two versions into one, yet.

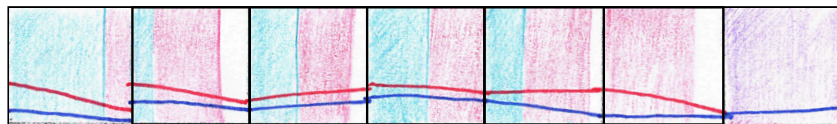
9.1 Ideation

The information the users were missing in the original interface (see Section 8.2) can be divided into two basic categories: (i) details about orders or activities, and (ii) time-dependent statistics of resource state.

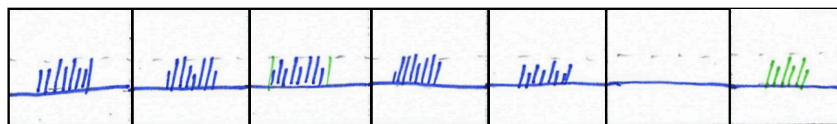
The information in the first category are quantities describing the execution of orders or activities, such as activity slack or order tardiness. In this case, the design process was trivial. We suggested adding new columns with appropriate filtering and sorting capabilities to the orders and activities tables.

On the other hand, for the statistics in the latter category, the change over time might be more interesting than the actual value. We suggested that the *proportion of tardy orders* processed on each day, the *orders schedulable at the beginning of a day* and the *resource utilization over time* could be shown in an interface divided into time segments—one segment representing a day.

Having this in mind, we started sketching out our ideas, the sketches of the most promising solutions are in Figure 9.1 (all sketches are part of the Appendix A). While the first visualization (Figure 9.1a) gives the user a broad overview of the resource utilization, the second one (Figure 9.1b) shows similar information in more detail.



(a) Background: proportion of tardy orders (blue: non-tardy, red: tardy, white: unused capacity, purple: not in operation). Lines: number of schedulable orders (blue: non-tardy, red: tardy).



(b) Utilization over time: vertical bar chart in each time segment, 24 hours on the x -axis, percentage of utilization on the y -axis.

Figure 9.1: Sketches of modified capacity analysis.

As the testing uncovered, the users prefer to see more information in a single interface. Thus, we also proposed a few changes to the overall information architecture (Figure 9.2). All the elements of the original interface (as introduced in Section 7.3) remained, but some were redesigned, and we moved the version tree and resource calendar to the main interface.

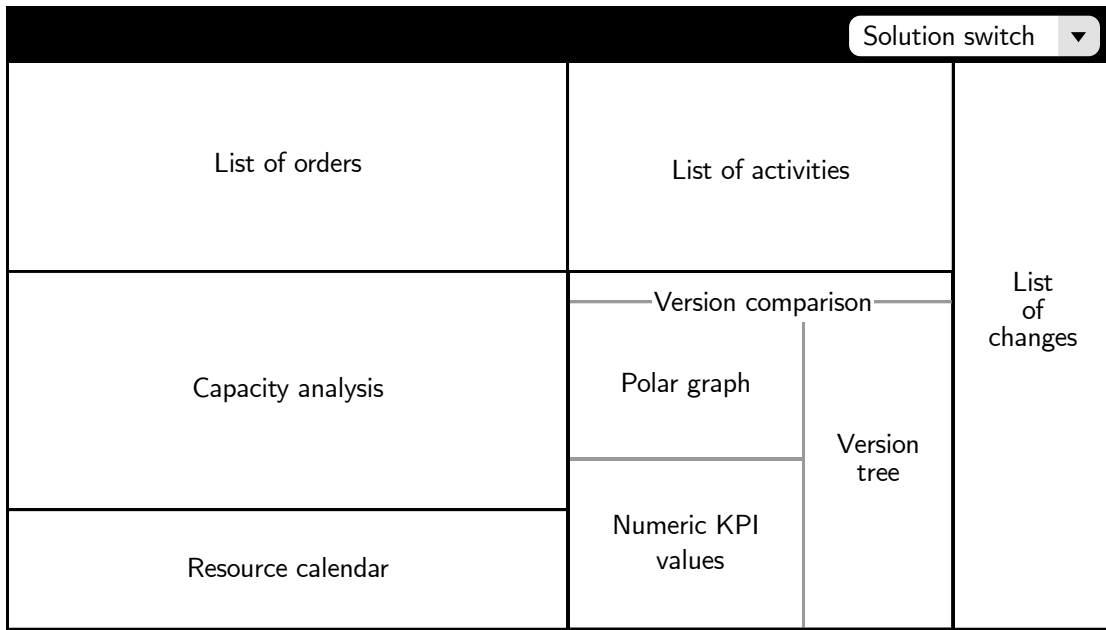


Figure 9.2: Redesigned main interface.

9.2 Clickable Mockup

We selected the most promising ideas and created a clickable mockup in Figma¹ (Figure 9.3). The prototype does not support rich interactions, but several actions are possible, such as clicking on tardy orders or switching between capacity analysis modes (Figure 9.4).

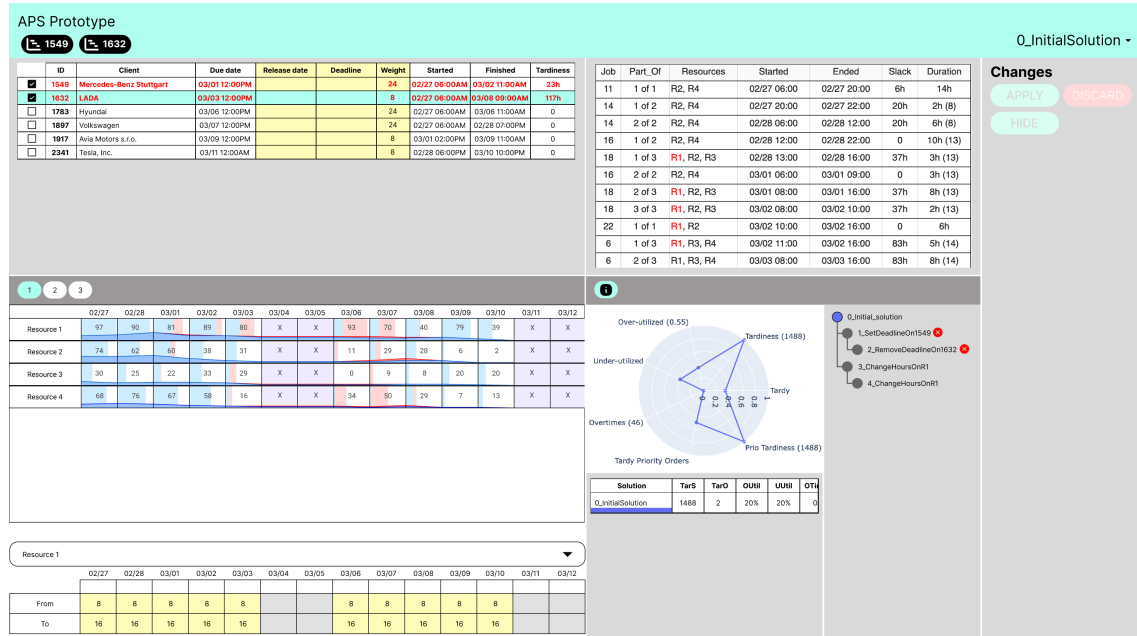
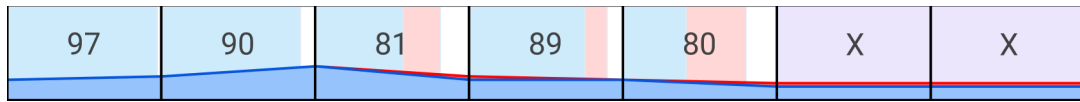
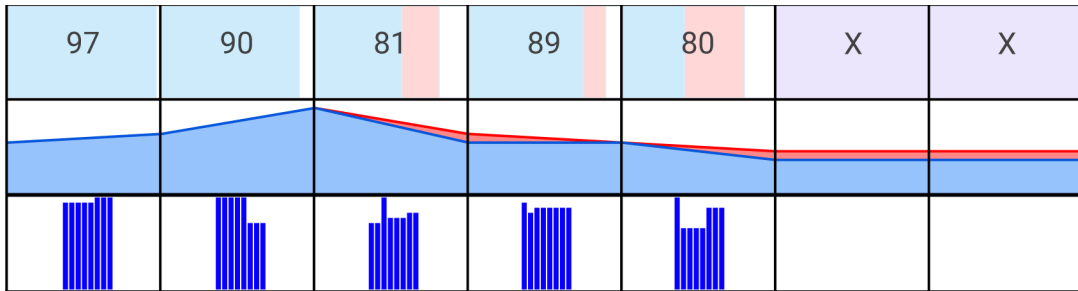


Figure 9.3: The main interface of the clickable mockup.

¹<https://www.figma.com/>



(a) Collapsed version.



(b) Expanded version.

Figure 9.4: Modified capacity analysis interface.

9.3 Clickable Mockup Testing

Having created the clickable mockup, we wanted to evaluate it and find out:

- (i) how intuitive are the new visualization components on the first use,
- (ii) what information are the users missing, and
- (iii) what changes in the information architecture are necessary.

We conducted *moderated in-person user testing* (the process is described in Section 3.3) of the clickable mockup with users who have a background in production scheduling ($N = 3$), their profiles are in Table 9.1. We prepared testing scenarios in which the users were asked to analyze the schedule and propose strategies to reduce the tardiness of orders (the exact formulations are in Appendix B). We asked them to verbalize their thoughts while analyzing the problem.

	Sex	Age	Current occupation	Experience
P1	M	56	manager	10 years
P2	M	51	sales manager, analyst	21 years
P3	M	34	developer	9 years

Table 9.1: Profiles of clickable mockup testers.

After the users proposed some strategies, the facilitator switched to a better solution. The users analyzed the schedule and verbally described actions that could help reduce tardiness. The sessions were recorded; a summary is part of the Appendix B.

How intuitive are the new visualization components on the first use?

The users had no problems with the interpretation of the detailed capacity analysis. However, two participants stated that the graphical representation of schedulable tasks is hard to interpret on the first use; they needed help understanding what the red part means. Only one participant could interpret it without any help. There was no problem with interpreting the polar graph with solution KPIs; the users could verbally summarize the changes and compare two solutions.

What information are the users missing?

During the testing, the users had several questions regarding the schedule that could not be easily answered in the current interface:

- What precedence relations are there?
- How will individual activities contribute to overall tardiness?
- In which order will the operations be processed on one resource?
- How will the execution of tardy orders be distributed on each day?
- What is the capacity (B^k) of the given resource?
- What are the operation hours of the given resource?
- What has been modified in this version?

What changes in the information architecture are necessary?

Overall, the evaluators rated their experience positively. All of them stated that the information was well-organized.

In summary, no changes to the overall information architecture are necessary. We could display more information in the capacity analysis interface (distribution of tardy orders on each day, capacity, operation hours, modified resources), in the tables (contribution to overall tardiness, modified orders), and in separate interfaces (precedence relations, operations processed on each resource). However, this testing did not provide much information about the usefulness of the new visualization components. We discuss sources of bias in moderated studies later in Section 10.3.

9.4 Coded Prototype

We transferred most of the ideas from the clickable mockup to the coded prototype, and added some more in reaction to user testing results. Compared to the first prototype, described in Chapter 7, more information is displayed in the main interface. Several interfaces remained untouched (*Gantt chart*, *changes*, *pairwise comparison*), and there are only minor changes in some interfaces (*orders and activities tables*, *resource calendar*). Other interfaces were redesigned (*capacity analysis*, *solution comparison* and *version tree*) or are entirely novel (*resource analytics*, *order analytics*). An overview of the new information architecture is given in Figure 9.5 and a screenshot of the main interface is presented in Figure 9.6.

9.4.1 Orders Table and Activities Table

Orders and activities tables (initially introduced in Subsection 7.3.1) have several new columns—we added *tardiness* (difference between actual finish date and due date) to the orders table, and *slack* (difference between the start time of activity and the finish time of its last predecessor) and *finish rate* (percentage of this activity finished before the execution is paused) to the activities table. Otherwise, these components remained the same.

Using the terminology of Munzner [57, pp. 225–234], slack values are highlighted using a colormap that is *ordered* (uses luminance channel that implies ordering), *sequential* (has only a single hue) and *segmented* (values are divided into five bins).

The finish rate is depicted as a *data bar*—the percentage of activity completion is mapped to the length of the bar. If the activity is interrupted before finishing—i.e., the finish rate is less than 100%—the bar is blue; otherwise, the bar is green.

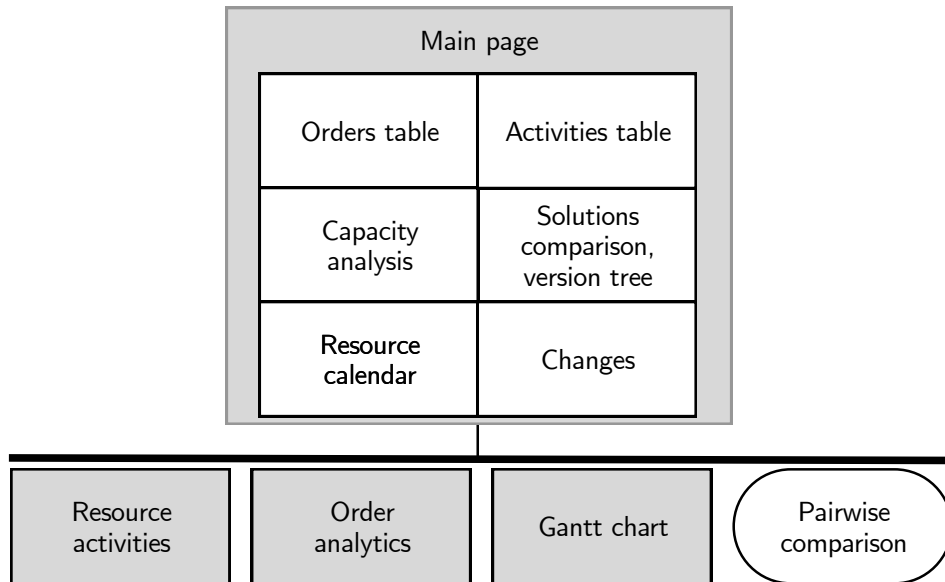


Figure 9.5: Scheme of information architecture. Gray rectangles are separate pages; rounded-corners-rectangles represent modal windows.

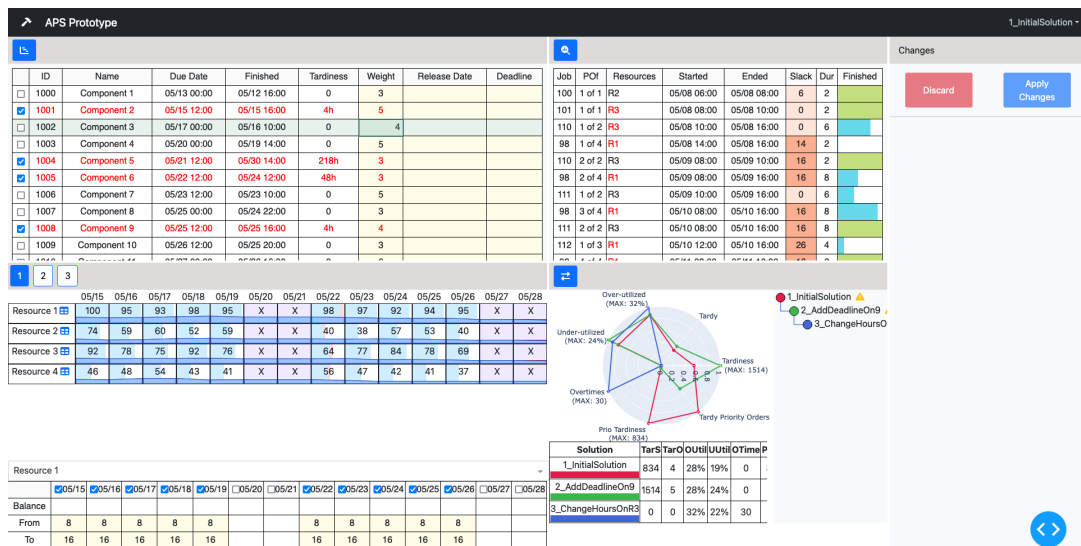


Figure 9.6: The main page screenshot.

9.4.2 Resource Calendar

Resource calendar (Figure 9.7, initially introduced in Subsection 7.3.6) was moved to the main interface, and a new row called *balance*, which contains information about the change in working hours, was added. This information is supported visually, and the cell is highlighted in orange if overtime hours were added and in green if working hours were reduced.

Resource 1														
	05/15	05/16	05/17	05/18	05/19	05/20	05/21	05/22	05/23	05/24	05/25	05/26	05/27	05/28
Balance	-2		2	1										
From	8	10	6	6	8			8	8	8	8	8		
To	14	18	16	15	16			16	16	16	16	16		

Figure 9.7: Resource calendar.

9.4.3 Capacity Analysis

Capacity analysis (initially introduced in Subsection 7.3.3) was redesigned using the ideas described in Section 9.1 and Section 9.2. It remains a *capacity buckets interface*, but the axes were switched—columns represent time and rows resources. For each resource, up to three rows can be shown (Figure 9.8). The first column contains a resource identification and a link to a page with details about the resource (further described in Subsection 9.4.5). Its purpose is to visually group and enclose information about one particular resource, and thus, it spans all related rows.

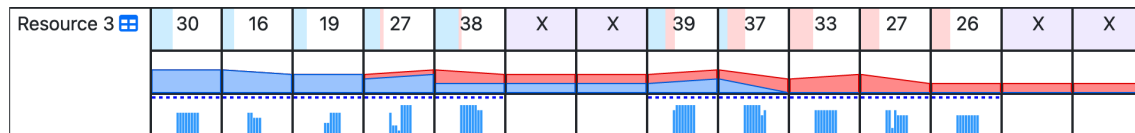


Figure 9.8: Resource in capacity analysis interface.

In the first row, the production capacity usage for the entire shift is presented as an integer (or “X” if the resource is not in operation on that day). Additionally, there is a *data bar* representation of this number with additional information about the proportion of non-tardy (blue) and tardy² (red) activities in the background.

The second row contains a graphical representation of the number of schedulable tasks. For each day, it displays the number of ready activities—i.e., the release time of the order has passed, and their precedences have been completed—at the beginning and the end of a day (typically 6:00 am and 10:00 pm, respectively). The number of ready tasks is divided between non-tardy (blue) and tardy (red) activities.

The third row displays information about the resource utilization over a day as miniaturized *vertical bar charts*. The *x*-axis is divided into 24 hours, and the *y*-axis shows the resource utilization. If the resource is in operation on a certain hour, but no activities are planned to be processed, there is an orange placeholder bar (slightly higher than a bar representing zero utilization) for that hour.

Users can select the level of detail, specifically the number of rows displayed for the resources. If they choose to display only one row for each resource, a miniaturized version of the *graphical representation of schedulable tasks* is shown together with the information about utilization, merging the first and second rows into one. Otherwise, the rows match the previous description. The component is scrollable, displaying only two weeks at a time.

9.4.4 Solution Comparison and Version Tree

Solution comparison (initially described in Subsection 7.3.2) was slightly changed. It now consists of two components—a polar chart and a table with numerical KPI values. The polar chart remains mostly the same, but based on the feedback from initial user testing, a fixed number of axes is displayed, even if no priority orders are selected. The numerical values of KPIs are presented in a table below the polar chart.

Version tree (originally introduced in Subsection 7.3.8) was moved to the main interface. The tree structure is still represented as a multilevel list, but lines connecting the nodes with their parents were added to make the relationships clearer to understand. Additionally, the checkboxes were replaced with checkable circles—if a version is selected, the circle is colored; otherwise, it is white.

These components are interconnected. When the user clicks on any circle in the tree, he selects or unselects the corresponding version. Only the values for the selected versions are

²By *tardy activities* we mean the activities that are processed after the due date of the order.

displayed in the polar chart and the KPIs table. Color mapping is shared between these three components—each version is assigned a color from a *categorical* colormap, and the color of the circle in the version tree, line in the polar chart, and line in the table all match.

9.4.5 Resource Activities

Resource activities interface (Figure 9.9) is a novel window that shows a table with activities processed on one particular resource in the planned order of execution. Its formatting is similar to other tables; tardy activities are highlighted in red.

Operations on Resource 1								
Job	Part Of	Order ID	Resources	Started	Ended	Slack	Duration	Finished
2	1 of 1	1000	R1, R2, R3, R4	05/15 08:00	05/15 10:00	8	2	
4	1 of 2	1000	R1, R2	05/15 08:00	05/15 16:00	8	8	
3	1 of 3	1000	R1, R2, R4	05/15 10:00	05/15 16:00	10	6	
11	1 of 2	1001	R1, R3	05/15 10:00	05/15 16:00	10	6	
4	2 of 2	1000	R1, R2	05/16 08:00	05/16 10:00	16	2	
11	2 of 2	1001	R1, R3	05/16 08:00	05/16 10:00	16	2	
3	2 of 3	1000	R1, R2, R4	05/16 08:00	05/16 16:00	16	8	
8	1 of 1	1000	R1, R2, R3	05/16 10:00	05/16 13:00	0	3	
14	1 of 1	1000	R1, R2, R4	05/16 10:00	05/16 15:00	24	5	
7	1 of 3	1000	R1, R2	05/16 13:00	05/16 16:00	27	3	
3	3 of 3	1000	R1, R2, R4	05/17 08:00	05/17 10:00	16	2	
7	2 of 3	1000	R1, R2	05/17 08:00	05/17 16:00	16	8	
5	1 of 1	1000	R1, R2, R3, R4	05/17 10:00	05/17 12:00	48	2	
15	1 of 2	1000	R1, R2, R4	05/17 10:00	05/17 16:00	0	6	
17	1 of 2	1001	R1, R3	05/17 12:00	05/17 16:00	60	4	
17	2 of 2	1001	R1, R3	05/18 08:00	05/18 09:00	16	1	
7	3 of 3	1000	R1, R2	05/18 08:00	05/18 11:00	16	3	
15	2 of 2	1000	R1, R2, R4	05/18 08:00	05/18 12:00	16	4	
18	1 of 1	1001	R1, R2, R3, R4	05/18 09:00	05/18 11:00	0	2	
19	1 of 2	1000	R1, R2, R3, R4	05/18 11:00	05/18 16:00	2	5	
19	2 of 2	1000	R1, R2, R3, R4	05/19 08:00	05/19 14:00	16	6	
23	1 of 2	1001	R1, R2, R3	05/22 09:00	05/22 16:00	94	7	
23	2 of 2	1001	R1, R2, R3	05/23 08:00	05/23 14:00	16	6	
21	1 of 1	1001	R1, R2, R3	05/23 12:00	05/23 15:00	121	3	
10	1 of 2	1001	R1, R2	05/23 14:00	05/23 16:00	0	2	
26	1 of 3	1002	R1, R3, R4	05/23 14:00	05/23 16:00	2	2	
16	1 of 2	1002	R1, R2, R3, R4	05/23 15:00	05/23 16:00	207	1	
10	2 of 2	1001	R1, R2	05/24 08:00	05/24 09:00	16	1	
16	2 of 2	1002	R1, R2, R3, R4	05/24 08:00	05/24 15:00	16	7	
26	2 of 3	1002	R1, R3, R4	05/24 08:00	05/24 16:00	16	8	

Figure 9.9: Resource activities screenshot.

9.4.6 Order Analytics

Order analytics (Figure 9.10) is a novel window that contains information about the execution of one particular order. Some of the design ideas are based on the strategies the users developed for improving the schedule, such as *setting release times for all orders but one to find out if it can be finished in time* or *finding an activity that is short, yet significantly contributes to tardiness, and moving it in time*.

The first strategy implies that the users may be interested in the execution of the order if other orders did not block it. Thus, we could show the earliest possible finish date (with the current setting of resources) and the latest release date to finish this order in time. The second strategy works with the contribution of one order to the tardiness. Obtaining this information is not straightforward; we use the precedence graph, and for every activity, we compute the tardiness of its last successor and compare it with the activity slack. Activities that have significant slack and significant successor tardiness are considered suspicious.

At the top of the page, there is textual information about (i) the earliest finish time, which is highlighted in red if the order cannot be finished in time with the current setting of resources, (ii) the scheduled finish time, and (iii) the latest release time to finish the order in time if other orders did not block the execution.

In the middle part of the page, there are two tables—the table on the left is identical to the activities table described in Subsection 9.4.1, the table on the right contains activities ranked by their improvement potential (minimum of slack and successor tardiness). This information is further supported by a simple *scatterplot* (Figure 9.11). Each point mark represents an activity, the position on the *x*-axis represents the improvement potential, and the position on the *y*-axis represents the duration of the activity.

9 Solution Prototype II

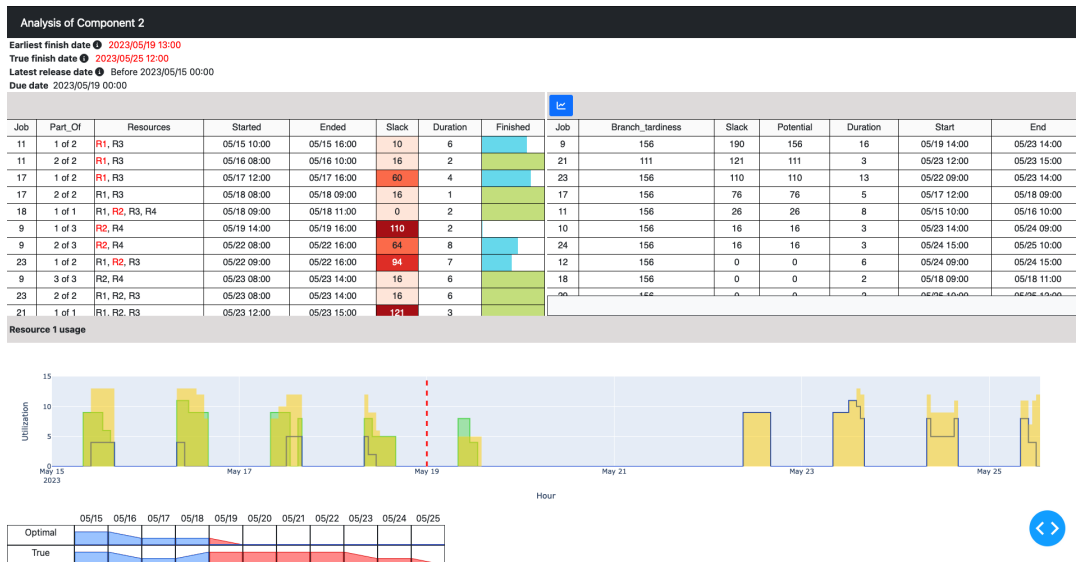


Figure 9.10: Order analytics screenshot.

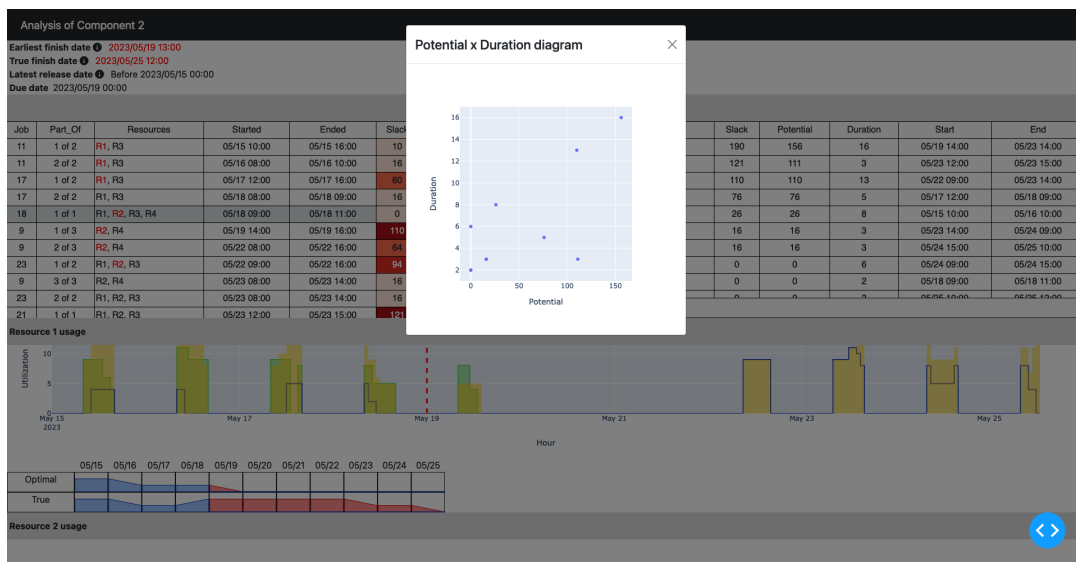


Figure 9.11: Modal window with Duration x Potential scatterplot.

At the bottom, information about the execution of this order on resources is shown. For each relevant resource, there is a graph with resource capacity utilization over time and a graphical representation of the number of schedulable tasks in this order.

Graph with resource capacity utilization contains four pieces of information: (i) when is the due date of this activity (red dashed vertical line), (ii) how would the resource be utilized if the execution of this order was not blocked by other orders (green), (iii) how utilized is the resource in the schedule (yellow), and (iv) how does this order contribute to the utilization of the resource (blue line).

10 Prototype II Evaluation

In this chapter, we present the testing of the second coded prototype (Chapter 9)—we introduce the goals, the methodology, and the results we obtained. In the discussion, we outline some directions for further evaluation. Scenarios we used for testing (in Czech) and a summary of the testing is part of Appendix B.

10.1 Methodology

We evaluated the new version of the fully-interactive UI with users to find out the following:

- (i) if the users can create a schedule matching the given criteria in this interface,
- (ii) how do they perceive the new interface,
- (iii) what information are they missing,
- (iv) how promising are the visualization components and interactions, and
- (v) what interface flaws do cause problems.

Before the testing, we prepared three problem instances, and for each problem instance, we prepared a scenario in which we defined the goals. The scenarios were focused on tardiness reduction. At the beginning of each testing session, we briefly described the components of the UI to the participant.

We conducted moderated user testing with experts in scheduling ($N = 3$); their profiles are in Table 10.1. All of them previously participated in clickable mockup testing and two of them in the first testing. We used the methodology we have already described in Section 3.3. The sessions were recorded; a summary is part of the Appendix B.

	Sex	Age	Current occupation	Experience
P1	M	51	sales manager, analyst	21 years
P2	M	34	developer	9 years
P3	M	56	manager	10 years

Table 10.1: Profiles of second coded prototype testers.

10.2 Results

We observed the users while they tried to improve a schedule to match certain criteria. The testing turned out to be more time-consuming than we had planned, and thus, we reduced the number of tasks the users were asked to accomplish to one.

Are the users able to create a schedule matching the given criteria?

All participants successfully finished the first scenario. They were also able to reduce the overtime hours in the schedule when they were asked to do so.

How do the users perceive the new interface?

The users' perception of the interface is positive; all participants appreciated that they have multiple views on data and each view provides them with some new information.

The users did not report any particular problem with understanding the design ideas we presented, nor any serious mismatch between their mental model of scheduling and the UI. One user stated that he expected activity to be one row in the activities table regardless of the interruptions, but he admitted that he is biased by the APS system he is used to. Otherwise, they could interpret the new visualizations correctly, although sometimes they asked for an additional explanation as they could not recall what some elements (colors, values, etc.) meant.

What information are the users missing?

They were missing better support for identifying the orders or activities that caused slack of other activities. One participant expressed a wish to have this information in the interface, and another searched for the intervening orders in the tables. One participant suggested that we could display the information about slack for all precedences, not just the last one, as this could help him backtrack the problem. They also analyzed the change in slack, but they had to remember the previous value.

How promising are the visualization components and interactions?

The usefulness of the visualization components remains hard to evaluate. However, we observed the evaluators using them, and thus, we evaluated their promise on that basis. There are several interfaces that the participants disregarded:

- Gantt chart (Subsection 7.3.7),
- graphical representation of schedulable tasks (Subsection 9.4.3),
- table with improvement potential and scatterplot with duration and improvement potential (Subsection 9.4.6).

Apparently, these components are not necessary to complete the tasks we assigned them.

On the other hand, we are confident that some visualization components were used and provided the users with information that helped them with decision-making. Namely, this is the case of:

- tables with orders and activities (Subsection 9.4.1),
- view with resource utilization over a day (Subsection 9.4.3),
- graphs with resource capacity utilization (Subsection 9.4.6).

Orders and *activities tables* were the first interfaces the users looked at when solving a task. They could identify the tardy orders because of their red color and estimate their tardiness, apparently by subtracting the due date from the finish date. For each order, they scanned the activities table and determined on which resources the order is processed and what activities are waiting for resources. The slack column seems to be particularly interesting to the users, or it is at least presumably visually salient, as the values are, unlike others, colored.

Resource utilization over a day demonstrably helped the users to identify the times when the resource is not processing any orders, and the capacity can be cut—they searched for the orange placeholder bars and reduced the working hours in case they found some. Two of the participants named it as a feature they liked.

Graphs with resource capacity utilization appear to lead the users to identify problematic resources correctly. One user also stated he likes that the graph contains information about the optimal execution.

Apart from the visualization components, the versioning and solution comparison systems appear to be very promising. All users agreed that the systems are well-designed. One participant suggested that we could improve the versioning by adding an option to “squash” multiple versions into one, as the progress toward a problem model is not so important to him.

What interface flaws cause problems?

Using the terminology of Nielsen [30], no usability catastrophes were detected, but several issues require attention:

- (i) editing the wrong problem model version (major),
- (ii) hard recovery from typos (major),
- (iii) incorrect reading of the resource utilization over a day graph (minor),
- (iv) ignoring the tardiness column (cosmetic).

All users encountered a problem with switching between versions; they sometimes *edited the wrong problem model*. Thus, we should make switching between solutions more user-friendly and, for example, show a “Switch to new solution” button after generating a new version. We could also implement “stash” and “unstash” features—if the user realized that he edited a wrong problem model, he could easily transfer the changes he made to the correct one.

The users also had a problem with *recovering from typos*, as the input fields do not allow deleting characters. Thus, they had to enter the input, read an error message, and type the input once again. This appears to be a bug in the library we use [58]. Regardless of that, it is important that the system should not be deployed with such usability issue.

We observed a problem with *reading the resource utilization over a day graph*. A tooltip with the corresponding time is shown when the user hovers over a placeholder. However, the users hovered over the incorrect placeholder and assumed that resource stopped being utilized later than it actually was. This is an implication of Fitts’s law (Equation (10.1)), which gives the relationship between the time (T) it takes a pointer to move to a particular target, the distance (D) to the target (D) and the width (w) of the target, a and b being constants that vary depending on the type of pointer [59].

$$T = a + b \cdot \log_2 \frac{2D}{w} \quad (10.1)$$

The placeholder is too small to be targeted, and during the final movement toward it, the users often overshoot and hovered beyond the target. We could solve this by showing the tooltip when the user hovers over a larger area with unused capacities instead of requiring to target the placeholder.

Participants apparently *estimated the tardiness by subtracting* the due date from the finish date. That is the less effective way to do so, given that there is also a column with the value just to the right of these two. However, they did not notice it and asked the moderator if it was present in the interface from the beginning. The value is expressed as a number, which does not attract users’ attention. Visual search is sequential unless the targets stand out [60, pp. 62–63]. Thus, if they scanned sequentially reading the table left-to-right, and their goal was to estimate the tardiness, they presumably continued to the next line when they subtracted the two values, taking no notice of the tardiness column. Validation of this hypothesis requires a study with an eye tracker, however. If we wanted to ensure that the tardiness column draws attention, we could add visual features that operate preattentively, i.e., they convey information before the viewer pays conscious attention [39, pp. 283–286], such as the colormap we already use for slack (described in Subsection 9.4.1).

10.3 Discussion

In summary, we created UI that is perceived positively by the experts in scheduling. Some of the visualizations seem to be promising, but some were disregarded by the users. Even

though we observed the users interacting with the scheduling system, it remains hard to evaluate the usefulness of the visualization components.

Additional studies would be required to uncover if the visualizations are useful. Eye tracking could be particularly interesting because it is applicable for examining how the users visually search the interface, allowing the researchers to see what exactly the users look for [61, p. 27]. However, it requires specialized equipment and software that are costly [61, p. 72]. Other types of quantitative studies seem to be applicable as well. For example, we could design a study to test if the users can create a schedule matching given criteria more quickly in the interface that contains certain visualizations than in a one that does not. However, the number of participants required to obtain statistically significant results may be large [62, p. 171].

We discussed the sources of bias for the first testing in Section 8.2. However, the sources of bias in in-person moderated studies are different. Observed studies are susceptible to *Hawthorne effect*. The participants may behave differently than in their natural environment because of the stress of being observed [27, pp. 39–40]. The sources of bias originating in the concrete test execution remain the same as for the first study—the external validity is limited, and the task may have been too straightforward compared to the reality the planners face. There is one additional source of bias for the moderated study—the sessions were moderated by the designer. It is almost impossible for moderators to remain objective when conducting a test for their own product. They have a too strong tendency to lead the participants in a direction they want the results to go [24, p. 46].

11 Conclusion and future work

Advanced Planning and Scheduling (APS) systems are complex applications that enable experts to create a production schedule, supporting the process with advanced mathematical algorithms. In this thesis, we focused on the problem of interaction design in such applications, which, as we discovered, remains unsolved in state-of-the-art systems.

Our aim for this work was to find out how can the user manipulate a schedule and keep track of the situation. We refined this aim into two main research questions:

- (i) What methods of interaction are suitable for manipulation of schedules?
- (ii) How to present the information about schedules to the users?

To tackle the problem, we applied the principles of human-centered design. To learn about the needs and expectations the users might have, we incorporated the experts on scheduling throughout the design process.

We surveyed the functionalities the scheduling systems have, their conceptual architecture, and mainly the forms user interfaces of scheduling systems take. Based on our research, we described the design problem using several UX design methods, including personas, scenarios, task analysis, and design requirements.

The essential part of this work was creating and testing UI prototypes. We created two types of prototypes: *coded*, implementing scheduling algorithms and allowing users to generate a schedule interactively, and a *clickable mockup*, enabling us to assess the information architecture and the clarity of the newly designed visualizations. With the exception of the first prototype, we generated design ideas based on the questions the users had about the schedule and the strategies they took to improve it.

We created novel visualization components depicting several characteristics of the schedule. The users have several views on the data available, and the views should provide them with some insights that help them make a decision.

Given that the APS system is a complex application with a non-linear workflow that is targeted at a specific population, its evaluation possesses certain challenges. Not only is there a relatively small number of users who have the domain expertise required to use the system, and only a few of them are accessible for testing, but also the workflows make the testing complicated. The problem the user solves is ill-defined, and it is up to him to redefine the model to achieve his end goal. We are also aware that it remains hard to evaluate if the visualizations are actually useful. Nevertheless, the testing should have been able to uncover mismatches between the users' mental model and the UI, and no such issues were found.

We discovered that effective manipulation of schedules could be achieved through adjusting the problem models. If we do not consider the decision-making process behind it, the interactions can be pretty straightforward. *Inline editing* of schedule parameters in tables and using the *forgiving format text fields* seem to be natural to the users.

The second research question remains open. We designed several novel visualization components that seem to be promising and discovered that the users prefer having multiple views on data, but we have not gathered sufficient evidence to generally prove if some types of presentation are superior to others. Additional quantitative studies would be required to uncover whether the visualizations (and which ones) are useful. Especially polar charts

seemed a promising solution for quickly comparing multiple schedules in overall metrics and metrics for a selected subset of orders.

11.1 Fulfillment of the Objectives

Here we describe how the objectives we had for this work were fulfilled.

Analyze the forms UI of APS systems take.

We surveyed the basic forms UI of APS systems take in Section 4.4, and in Section 4.5 we presented the use of the components in a commercial APS system.

Create a first prototype with typical components of APS system.

We created the first prototype of a scheduling system that uses basic components and allows interactions with a scheduling algorithm. The model of the scheduling problem we used to simulate the capabilities of a scheduling system was described in Chapter 6. Implementation of the prototype, and mainly the UI components, was introduced in Chapter 7.

Test the first prototype with experts in scheduling and identify their needs.

We described the exploratory testing of the first prototype in Chapter 8.

Design novel visualization components and implement them into the prototype.

On the basis of previous evaluation, we identified several opportunities, which we transferred into design ideas described in Section 9.1. We created a clickable mockup containing the newly designed visualization components (Section 9.2), which we tested with the users (Section 9.3). We improved the design ideas and implemented the new visualizations into a second version of the interactive prototype (Section 9.4).

Evaluate the prototype with professional users of APS systems.

We evaluated the fully-interactive UI with scheduling experts. The evaluation process and its results were described in Chapter 10.

11.2 Future Work

It seems to us that supporting the scheduling process with interactive visualizations is very promising. However, additional quantitative studies would be required to evaluate if the visualizations we designed are useful. For example, eye tracking could help us reveal which visualizations do the users look at during the decision-making. We could also design a study to test if the users can create a schedule matching given criteria more quickly or on less unsuccessful attempts, comparing an interface that contains certain visualizations to one that does not, on a set of benchmark instances.

Finally, the ideas we presented in this work, ideally after proving their effectiveness, could be implemented into a real APS system.

Bibliography

1. RITTEL, Horst W. J. and WEBBER, Melvin M. Dilemmas in a general theory of planning. *Policy sciences*. 1973, vol. 4, no. 2, pp. 155–169. Available from DOI: 10.1007/BF01405730.
2. MAN, Johannes Cornelis de and STRANDHAGEN, Jan Ola. Spreadsheet application still dominates enterprise resource planning and advanced planning systems. *Ifac-Papersonline*. 2018, vol. 51, no. 11, pp. 1224–1229. Available from DOI: 10.1016/j.ifacol.2018.08.423.
3. HVOLBY, Hans-Henrik and STEGER-JENSEN, Kenn. Technical and industrial issues of Advanced Planning and Scheduling (APS) systems. *Computers in Industry*. 2010, vol. 61, no. 9, pp. 845–851. Available from DOI: 10.1016/j.compind.2010.07.009.
4. JÜNGEN, FJ and KOWALCZYK, W. An intelligent interactive project management support system. *European Journal of Operational Research*. 1995, vol. 84, no. 1, pp. 60–81. Available from DOI: 10.1016/0377-2217(94)00318-7.
5. MCKAY, Kenneth N. and WIERS, Vincent C.S. The human factor in planning and scheduling. In: *Handbook of Production Scheduling*. Springer, 2006, pp. 23–57. Available from DOI: 10.1007/0-387-33117-4_2.
6. FRAMINAN, Jose M. et al. *Manufacturing scheduling systems*. Springer, 2014. ISBN 978-1-4471-6272-8.
7. JO, Jaemin et al. LiveGantt: Interactively visualizing a large manufacturing schedule. *IEEE transactions on visualization and computer graphics*. 2014, vol. 20, no. 12, pp. 2329–2338. Available from DOI: 10.1109/TVCG.2014.2346454.
8. KLEIN, Gary. Naturalistic decision making. *Human factors*. 2008, vol. 50, no. 3, pp. 456–460. Available from DOI: 10.1518/001872008X288385.
9. WIERS, Vincent C.S. *Human computer interaction in production scheduling*. Citeseer, 1997. Available from DOI: 10.6100/IR495263.
10. RASMUSSEN, Jens. Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE transactions on systems, man, and cybernetics*. 1983, no. 3, pp. 257–266. Available from DOI: 10.1109/TSMC.1983.6313160.
11. CEGARRA, Julien. A cognitive typology of scheduling situations: A contribution to laboratory and field studies. *Theoretical Issues in Ergonomics Science*. 2008, vol. 9, no. 3, pp. 201–222. Available from DOI: 10.1080/14639220601095379.
12. KERR, R. M. and EBSARY, R. V. Implementation of an expert system for production scheduling. *European journal of operational research*. 1988, vol. 33, no. 1, pp. 17–29. Available from DOI: 10.1016/0377-2217(88)90250-0.
13. FOX, Mark S. Constraint-guided scheduling—A short history of research at CMU. *Computers in Industry*. 1990, vol. 14, no. 1-3, pp. 79–88. Available from DOI: 10.1016/0166-3615(90)90107-Z.

14. MEIGNAN, David et al. A review and taxonomy of interactive optimization methods in operations research. *ACM Transactions on Interactive Intelligent Systems (TiiS)*. 2015, vol. 5, no. 3, pp. 1–43. Available from DOI: 10.1145/2808234.
15. LIU, Jie et al. Understanding the relationship between interactive optimisation and visual analytics in the context of prostate brachytherapy. *IEEE transactions on visualization and computer graphics*. 2017, vol. 24, no. 1, pp. 319–329. Available from DOI: 10.1109/TVCG.2017.2744418.
16. LIU, Jie et al. Supporting the problem-solving loop: Designing highly interactive optimisation systems. *IEEE Transactions on Visualization and Computer Graphics*. 2020, vol. 27, no. 2, pp. 1764–1774. Available from DOI: 10.48550/arXiv.2009.03163.
17. KAPLAN, Kate. *Complex Application Design: A 5-Layer Framework* [online]. Nielsen Norman Group, 2020-08-09. [visited on 2023-03-30]. Available from: <https://www.nngroup.com/articles/complex-application-design-framework/>.
18. KEIM, Daniel et al. *Mastering the information age solving problems with visual analytics*. Eurographics Association, 2010. ISBN 978-3-9056-7377-7.
19. KAPLAN, Kate. *Complex Application 101* [online]. Nielsen Norman Group, 2020-09-04. [visited on 2023-03-30]. Available from: <https://www.youtube.com/watch?v=NHevdlqo0aQ>.
20. NATHAN, Aaron. *Designing for Experts Without Being An Expert* [online]. Bentley User Experience Center, 2018-10. [visited on 2023-04-27]. Available from: <https://www.bentley.edu/centers/user-experience-center/designing-experts-without-being-expert>.
21. ISO 9241-210:2019. *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. 2019.
22. GOTHELF, Jeff. *Lean UX*. O’Reilly Media, Inc., 2013. ISBN 978-1-4493-1165-0.
23. WALKER, Miriam et al. High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. In: *Proceedings of the human factors and ergonomics society annual meeting*. Sage Publications Sage CA: Los Angeles, CA, 2002, vol. 46, pp. 661–665. No. 5. Available from DOI: 10.1177/154193120204600513.
24. RUBIN, Jeffrey and CHISNELL, Dana. *Handbook of usability testing: how to plan, design and conduct effective tests*. John Wiley & Sons, 2008. ISBN 978-0-4701-8548-3.
25. RAMASWAMY, Sara. *The Wizard of Oz Method in UX* [online]. Nielsen Norman Group, 2022-11-20. [visited on 2023-05-04]. Available from: <https://www.nngroup.com/articles/wizard-of-oz/>.
26. GOODMAN, Elizabeth and KUNIAVSKY, Mike. *Observing the user experience: A practitioner’s guide to user research*. Elsevier, 2012. ISBN 978-0-1238-4869-7.
27. LAZAR, Jonathan et al. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017. ISBN 978-0-1280-9343-6.
28. VIRZI, Robert A. Refining the test phase of usability evaluation: How many subjects is enough? *Human factors*. 1992, vol. 34, no. 4, pp. 457–468. Available from DOI: 10.1177/001872089203400407.
29. CARPENDALE, Sheelagh. Evaluating information visualizations. *Information visualization: Human-centered issues and perspectives*. 2008, pp. 19–45.

30. NIELSEN, Jakob. *Severity Ratings for Usability Problems* [online]. Nielsen Norman Group, 1994-11-01. [visited on 2023-05-16]. Available from: <https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/>.
31. KREIPL, Stephan and PINEDO, Michael. Planning and scheduling in supply chains: an overview of issues in practice. *Production and Operations management*. 2004, vol. 13, no. 1, pp. 77–92. Available from DOI: 10.1111/j.1937-5956.2004.tb00146.x.
32. STADTLER, Hartmut. Supply chain management: an overview. In: *Supply chain management and advanced planning*. Springer, 2015, pp. 3–28. Available from DOI: 10.1007/978-3-540-74512-9_2.
33. KJELLSDOTTER IVERT, Linea and JONSSON, Patrik. Problems in the onward and upward phase of APS system implementation: Why do they occur? *International Journal of Physical Distribution & Logistics Management*. 2011, vol. 41, no. 4, pp. 343–363. Available from DOI: 10.1108/09600031111131922.
34. AYTUG, Haldun et al. A review of machine learning in scheduling. *IEEE Transactions on Engineering Management*. 1994, vol. 41, no. 2, pp. 165–171. Available from DOI: 10.1109/17.293383.
35. PINEDO, Michael. *Scheduling*. Vol. 29. Springer, 2012. ISBN 978-1-4614-2361-4.
36. HIGGINS, Peter G. Interaction in hybrid intelligent scheduling. *International Journal of Human Factors in Manufacturing*. 1996, vol. 6, no. 3, pp. 185–203. Available from DOI: 10.1002/(SICI)1522-7111(199622)6:3<185::AID-HFM1>3.0.CO;2-6.
37. LAUBHEIMER, Page. *Data tables: four major user tasks* [online]. Nielsen Norman Group, 2022-04-03. [visited on 2023-01-05]. Available from: <https://www.nngroup.com/articles/data-tables/>.
38. PINEDO, Michael. Design and implementation of scheduling systems: Basic concepts. In: *Scheduling*. Springer International Publishing, 2022, pp. 467–491. Available from DOI: 10.1007/978-3-031-05921-6_17.
39. TIDWELL, Jenifer. *Designing interfaces: Patterns for effective interaction design*. O’Reilly Media, Inc., 2010. ISBN 978-1-4493-7970-4.
40. COOPER, Alan et al. *About face: the essentials of interaction design*. John Wiley & Sons, 2014. ISBN 978-1-1187-6657-6.
41. GIBBONS, Sarah. *UX stories communicate designs* [online]. Nielsen Norman Group, 2017-01-15. [visited on 2022-12-21]. Available from: <https://www.nngroup.com/articles/ux-stories/>.
42. ROSALA, Maria. *Task analysis: support users in achieving their goals* [online]. Nielsen Norman Group, 2020-09-20. [visited on 2022-12-20]. Available from: <https://www.nngroup.com/articles/task-analysis/>.
43. ARTIGUES, Christian et al. The resource-constrained project scheduling problem. In: *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013, pp. 21–35. Available from DOI: 10.1002/9780470611227.ch1.
44. LABORIE, Philippe. *Industrial project and machine scheduling with constraint programming*. 2021. Available from DOI: 10.13140/RG.2.2.30470.70726. [Conference presentation].

45. GRAHAM, Ronald Lewis et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: *Annals of discrete mathematics*. Elsevier, 1979, vol. 5, pp. 287–326. Available from DOI: 10.1016/S0167-5060(08)70356-X.
46. HALL, Nicholas G. and POSNER, Marc E. Generating experimental data for computational testing with machine scheduling applications. *Operations Research*. 2001, vol. 49, no. 6, pp. 854–865. Available from DOI: 10.1287/opre.49.6.854.10014.
47. KRETER, Stefan et al. Using constraint programming for solving RCPSP/max-cal. *Constraints*. 2017, vol. 22, no. 3, pp. 432–462. Available from DOI: 10.1007/s10601-016-9266-6.
48. DECHTER, Rina; COHEN, David, et al. *Constraint processing*. Morgan Kaufmann, 2003. ISBN 978-1-5586-0890-0.
49. IBM. *IBM ILOG CP Optimizer* [online]. IBM. [visited on 2023-05-21]. Available from: <https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-cp-optimizer>.
50. PLOTLY. *Dash Python User Guide* [online]. Plotly. [visited on 2023-05-17]. Available from: <https://dash.plotly.com/>.
51. PLOTLY. *Deploying Dash Apps* [online]. Plotly. [visited on 2023-05-17]. Available from: <https://dash.plotly.com/deployment>.
52. CATALOGUE, The Data Visualisation. *Radar Chart* [online]. [visited on 2023-04-29]. Available from: https://datavizcatalogue.com/methods/radar_chart.html.
53. HARTMANN, Sönke. Packing problems and project scheduling models: an integrating perspective. *Journal of the Operational Research Society*. 2000, vol. 51, no. 9, pp. 1083–1092. Available from DOI: 10.2307/254229.
54. CSÉBFALVI, A. A theoretically correct resource usage visualization for the resource-constrained project scheduling problem. *Iran University of Science & Technology*. 2012, vol. 2, no. 2, pp. 173–181.
55. BRUUN, Anders et al. Let your users do the testing: a comparison of three remote asynchronous usability testing methods. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009, pp. 1619–1628. Available from DOI: 10.1145/1518701.1518948.
56. COZBY, Paul C. et al. *Methods in behavioral research*. 2012. ISBN 978-0-0778-6189-6.
57. MUNZNER, Tamara. *Visualization analysis and design*. CRC Press, 2014. ISBN 978-1-4665-0891-0.
58. CMPCTRL. *[BUG] Dash editable DataTable issues. Issue #2018* [online]. Github, 2022-04-18. [visited on 2023-05-16]. Available from: <https://github.com/plotly/dash/issues/2018>.
59. BUDIU, Raluca. *Fitts's Law and Its Applications in UX* [online]. Nielsen Norman Group, 2022-07-31. [visited on 2023-05-16]. Available from: <https://www.nngroup.com/articles/fitts-law/>.
60. JOHNSON, Jeff. *Designing with the Mind in Mind*. Morgan Kaufmann, 2014. ISBN 978-0-1240-7914-4.
61. BERGSTROM, Jennifer Romano and SCHALL, Andrew. *Eye tracking in user experience design*. Elsevier, 2014. ISBN 978-0-1240-8138-3.
62. MACKENZIE, I. Scott. *Human-computer interaction: An empirical research perspective*. 2013. ISBN 978-0-1240-5865-1.

Appendix

Here, we provide an overview of the electronic appendices.

A Design Process Documentation

This part of the appendix contains additional documentation for the design process, including:

- HTA diagram and plans,
- interface sketches with handwritten notes (in Czech).

B User Testing Documentation

This part of the appendix contains additional documentation for user testing, including

- the document with instructions we sent to the users for the first study (in Czech),
- summary of the first testing,
- scenarios for the clickable mockup testing (in Czech),
- summary of the clickable mockup testing,
- scenarios for the second coded prototype testing (in Czech),
- summary of the second coded prototype testing.

C Source Code

This part of the appendix contains source codes for the prototype. Instructions are part of the `README.md` file in the project root directory.