

# Analysis and Optimization of Mobile Business Processes

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften  
(Dr. rer. nat.)

durch die  
Fakultät für Wirtschaftswissenschaften  
Institut für Informatik und Wirtschaftsinformatik  
Universität Duisburg-Essen  
Campus Essen

vorgelegt von  
**Dipl.-Inf. Thomas Richter**  
geboren in Leipzig

Essen (2012)

Tag der mündlichen Prüfung: 4. Juni 2012

Erstgutachter: Prof. Dr. Volker Gruhn

Zweitgutachter: Prof. Dr. Stefan Eicker

# Contents

<b>Acknowledgements</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Abbreviations</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Business Processes and Mobility . . . . .	1
1.1.1 Business Processes . . . . .	1
1.1.2 Mobile Business Processes . . . . .	3
1.1.3 Workflow Management of Mobile Business Processes .	5
1.1.4 Performing Business Change . . . . .	8
1.2 Identified Problem and Contribution of this Work . . . . .	10
1.2.1 Modeling and Simulation of Mobile Business Processes	10
1.2.2 Workforce Scheduling . . . . .	11
1.2.3 Solution Overview . . . . .	12
<b>2 Domain Model of Mobility</b>	<b>13</b>
2.1 Utility Industry Process Examples . . . . .	13
2.1.1 Asset Inspection . . . . .	14
2.1.2 Switched Power Line Processes . . . . .	14
2.1.3 Damage Search and Repair . . . . .	15

2.2	Abstract Model of Mobile Processes . . . . .	17
2.2.1	Assignment of Mobile Work . . . . .	18
2.2.2	Performance of Mobile Work . . . . .	19
2.3	Entities of Mobile Work . . . . .	20
2.4	State Models of Entities . . . . .	24
2.4.1	Mobile Workers . . . . .	24
2.4.2	Mobile Tasks . . . . .	26
2.4.3	Mobile Cases . . . . .	28
2.5	Optimization Objectives . . . . .	30
2.5.1	Worker Related Costs . . . . .	31
2.5.2	Travel Related Costs . . . . .	32
2.5.3	Case Related Costs . . . . .	33
2.5.4	Optimization Parameters of Mobile Environments . . .	35
2.6	Chapter Summary . . . . .	38
<b>3</b>	<b>Simulation of Mobility</b>	<b>39</b>
3.1	Introduction to Dynamic Analysis and Simulation . . . . .	40
3.1.1	Problems Addressed . . . . .	40
3.1.2	General Execution of Simulations . . . . .	41
3.2	Colored Petri Net Domain Model of Mobility . . . . .	42
3.2.1	Entities of Mobile Work . . . . .	42
3.2.2	Model Overview . . . . .	45
3.2.3	Loading and Initializing Simulation Data . . . . .	47
3.2.4	Generating Cases and Tasks . . . . .	51
3.2.5	Scheduling the Workforce . . . . .	52
3.2.6	State Model of Mobile Workers . . . . .	54
3.2.7	Postprocessing the Simulation Results . . . . .	59

3.3	Composition of Business Processes . . . . .	60
3.3.1	Process Modeling Interface . . . . .	60
3.3.2	Control Structures . . . . .	63
3.3.3	Process Initialization . . . . .	67
3.3.4	Process Finalization . . . . .	68
3.3.5	Section Summary . . . . .	69
3.4	Reducing the Modeling Effort . . . . .	69
3.4.1	Modeling Language Requirements . . . . .	72
3.4.2	Simple Mobile Process Language Overview . . . . .	75
3.4.3	SMPL Elements and Semantics . . . . .	78
3.4.4	Transformations of SMPL Models to the CPN Domain Model . . . . .	82
3.5	Chapter Summary . . . . .	86
<b>4</b>	<b>Mobile Workforce Scheduling</b>	<b>87</b>
4.1	Introduction . . . . .	87
4.2	Scheduling Paradigms . . . . .	88
4.2.1	Planned Workforce Scheduling . . . . .	89
4.2.2	Dynamic Workforce Scheduling . . . . .	91
4.3	Related Scheduling Problems . . . . .	96
4.3.1	The Vehicle Routing Problem . . . . .	97
4.3.2	The VRP with Time Windows . . . . .	99
4.3.3	The Resource-Constrained Project Scheduling Problem	101
4.3.4	The RCPSP with Preemption . . . . .	103
4.3.5	The RCPSP with Multi-Projects . . . . .	105
4.4	The MWSP-MP . . . . .	107
4.4.1	Introduction . . . . .	107
4.4.2	Formulation of the MWSP-MP . . . . .	109

4.5	Solution Methods of Scheduling Problems . . . . .	112
4.5.1	Assignment Methods . . . . .	113
4.5.2	Construction Methods . . . . .	113
4.5.3	Improvement Methods . . . . .	114
4.5.4	Meta-heuristics . . . . .	115
4.6	Solution Methods for the MWSP-MP . . . . .	116
4.6.1	Feasibility Criteria for Schedules and Solutions . . . .	117
4.6.2	Neighborhood Operator INSERT . . . . .	118
4.6.3	Neighborhood Operator REMOVE . . . . .	126
4.6.4	Creating Start Solutions . . . . .	129
4.7	Chapter Summary . . . . .	136
<b>5</b>	<b>Validation</b>	<b>137</b>
5.1	Subject of Study . . . . .	137
5.1.1	ENSO—A German Gas and Power Supply . . . . .	137
5.1.2	Processes and Cases . . . . .	140
5.2	Workforce Scheduling Methods . . . . .	146
5.3	Simulation of Scenarios . . . . .	149
5.3.1	Method . . . . .	149
5.3.2	Scenarios . . . . .	150
5.3.3	Simulation Results and Discussions . . . . .	151
5.3.4	Section Summary . . . . .	158
5.4	Tool Support . . . . .	163
5.4.1	Manipulating Simulation Data . . . . .	163
5.4.2	Modeling Business Processes . . . . .	164
5.5	Chapter Summary . . . . .	166
<b>6</b>	<b>Conclusion</b>	<b>167</b>
6.1	Scientific Contributions . . . . .	167
6.2	Future Research . . . . .	168
	<b>Symbols</b>	<b>171</b>
	<b>Bibliography</b>	<b>175</b>

# Acknowledgements

I would like to thank all who supported me during the completion of the thesis. This thesis would not have been possible without the valuable feedback from my first supervisor, Prof. Dr. Volker Gruhn, and the freedom he gave me to pursue my interests. I would also like to thank Prof. Dr. Stefan Eicker for agreeing to be my second supervisor.

I'm really grateful for all the helpful comments and hints I received from my colleagues at the institute. Especially I would like to thank Asvin Goel for his valuable input during our discussions about the formulation of the MWSP-MP. I would also like to thank Carolin Ulbricht who developed a preliminary version of the scheduling software in her master's thesis (see Section 5.2), Alex Klebeck who developed the communication interface of the scheduler in his master's thesis (see Section 3.2.5), and Steffen Schulz as well as Sebastian Neudert who developed the simulation control tool during their master's theses (see Section 5.4). I would further like to thank Katharina König for proofreading this thesis.

My special thanks go to Sven Böttcher and Dr. Steffen Heine of ENSO who provided the validation data and were never tired answering my questions.

Last but not least I would like to thank my beloved wife Gwendolyn who often cut back her own interests to let me complete this thesis.





# Abstract

Mobility of workers and business processes rapidly gains the attention of businesses and business analysts. A wide variety of definitions exists for mobile business processes. This work considers a type of business processes concerned with the maintenance of distributed technical equipment as, e.g., telecommunication networks, utility networks, or professional office gear. Executing the processes in question, workers travel to the location where the equipment is situated and perform tasks there. Depending on the type of activities to be performed, the workers need certain qualifications to fulfill their duty. Especially in network maintenance processes, activities are often not isolated but depend on the parallel or subsequent execution of other activities at other locations. Like every other economic activity, the outlined mobile processes are under permanent pressure to be executed more efficiently. Since business process reengineering (BPR) projects are the common way to achieve process improvements, business analysts need methods to model and evaluate mobile business processes.

Mobile processes challenge BPR projects in two ways: (i) the process attributes introduced by mobility (traveling, remote synchronization, etc.) complicate process modeling, and (ii) these attributes introduce process dynamics that prevent the straightforward prediction of BPR effects. This work solves these problems by developing a modeling method for mobile processes. The method allows for simulating mobile processes considering the mobility attributes while hiding the complexity of these attributes from the business analysts modeling the processes.

Simulating business processes requires to assign activities to workers, which is called scheduling. The spatial distribution of activities relates scheduling to routing problems known from the logistics domain. To provide the simulator with scheduling capabilities the according Mobile Workforce Scheduling Problem with Multitask-Processes (MWSP-MP) is introduced and analyzed in-depth. A set of neighborhood operators was developed to allow for the application of heuristics and meta-heuristics to the problem. Furthermore, methods for generating start solutions of the MWSP-MP are introduced.

The methods introduced throughout this work were validated with real-world

data from a German utility. The contributions of this work are a reference model of mobile work, a business domain independent modeling method for mobile business processes, a simulation environment for such processes, and the introduction and analysis of the Mobile Workforce Scheduling Problem with Multitask-Processes.

# Abbreviations

<b>AD</b>	Activity Diagram
<b>AMP</b>	Adaptive Memory Programming
<b>AON</b>	Activity-On-Node (network)
<b>AQ</b>	Additional Qualifications
<b>ARMS</b>	Automated Resource Management System
<b>BFS</b>	Breadth First Search
<b>BPR</b>	Business Process Reengineering
<b>BPMN</b>	Business Process Modeling Notation
<b>BPS</b>	Business Process Simulation
<b>CCS</b>	Calculus of Communicating Systems
<b>CD</b>	Closest Depot
<b>CEMF-DO</b>	Common Enterprise Modeling Framework for Distributed Organizations
<b>CONWIP</b>	CONstant Work In Progress
<b>CPN</b>	Colored Petri Net
<b>DARP</b>	Dynamic Asset Replacement Planning
<b>DFS</b>	Depth First Search
<b>EATE</b>	Equal Average Travel Effort
<b>EAWR</b>	Equal Asset-Worker Ratio
<b>EBNF</b>	Extended Backus-Naur Form
<b>EM</b>	Enterprise Modeling

<b>EPC</b>	Event-driven Process Chain
<b>EQD</b>	Equal Qualification Distribution
<b>ERP</b>	Enterprise Resource Planning
<b>FMC-PND</b>	Fundamental Modeling Concept Petri Net Diagram
<b>FTL</b>	Full-Truckload
<b>FTL PDP</b>	Full-Truckload Pickup and Delivery Problem
<b>GPDP</b>	General Pickup and Delivery Problem
<b>GPS</b>	Global Positioning System
<b>GVRP</b>	General Vehicle Routing Problem
<b>ILS</b>	Iterated Local Search
<b>KPI</b>	Key Process Indicator
<b>LD</b>	Location of the Depot
<b>LNS</b>	Large Neighborhood Search
<b>MoSim</b>	Mobile Organization Sim Control
<b>MWSP-MP</b>	Mobile Workforce Scheduling Problem with Multitask-Processes
<b>OCL</b>	Object Constraint Language
<b>OOP</b>	Object Oriented Programming
<b>PDA</b>	Personal Digital Assistant
<b>PDP</b>	Pickup and Delivery Problem
<b>PDPTW</b>	Pickup and Delivery Problem with Time Windows
<b>PEF</b>	Process Execution Frequency
<b>PN</b>	Petri Net
<b>PRCPSP</b>	Preemptive Resource-Constrained Project Scheduling Problem
<b>QDS</b>	Qualification Dependent Scheduling
<b>RCPSP</b>	Resource-Constrained Project Scheduling Problem
<b>RCMPSP</b>	Resource-Constrained Multi-Project Scheduling Problem

<b>REQ</b>	Requirement
<b>SHS</b>	Scheduler Heuristic Selection
<b>SL</b>	Start Location
<b>SMPL</b>	Simple Mobile Process Language
<b>TCP</b>	Transmission Control Protocol
<b>TNR</b>	Total Number of Regions
<b>TSP</b>	Traveling Salesman Problem
<b>UEML</b>	Unified Enterprise Modeling Language
<b>UML</b>	Unified Modeling Language
<b>UML AD</b>	Unified Modeling Language Activity Diagram
<b>VNS</b>	Variable Neighborhood Search
<b>VRP</b>	Vehicle Routing Problem
<b>VRPTW</b>	Vehicle Routing Problem with Time Windows
<b>WATE</b>	Weighted Average Travel Effort
<b>YAWL</b>	Yet Another Workflow Language



# Chapter 1

## Introduction

### 1.1 Business Processes and Mobility

During the last decades economic reality was constantly characterized by an increasing dynamic. Organizations must respond timely to the changing economic conditions; and, therefore, a static view of their business does not meet their needs. The increasing availability of wireless data communication and mobile electronic devices introduces even more dynamic aspects to business execution. In this section we will introduce the terms *business process* and *mobile business process* as well as the implications of mobility to the execution of business processes.

#### 1.1.1 Business Processes

One way for an enterprise to understand its business processes is to consider the business functions in a holistic manner. Considering an organization's structure both static and dynamic aspects strike. While the structural (static) organization of an enterprise covers the hierarchical distribution of duties and responsibilities, the operational (dynamic) organization covers the actual assignment of the work items to the employees [34].

Early business improvement efforts concentrated on the local optimization of single business functions in an isolated way. These developments were enabled by science and technology, namely the upgrowth of electronic data processing and communication technology. The outcomes of complex business tasks are fundamentally determined by the interaction of the business functions. Especially the effort necessary to coordinate isolated business functions can lead to suboptimal business results [8]. The consideration of business processes by enterprises essentially increased during the 1980s after the topic had already been covered by researchers for years. Different perceptions of the

term *business process* emerged. The following definition given by Becker et al. in [8] is affected by economic research:

“A process is a self-contained, chronological, and logical sequence of activities necessary to process an economically relevant object.”

A more technical definition is given by van der Aalst and van Hee in [124]:

“A process consists of a number of tasks that need to be carried out and a set of conditions that determine the order of the tasks. A process can also be called a procedure. A task is a logical unit of work that is carried out as a single whole by one resource. A resource is the generic name for a person, machine, or group of persons or machines that can perform specific tasks. This does not always mean to say that the resource necessarily carries out the task independently, but that it is responsible for it.”

This process definition given by van der Aalst and van Hee will be the basis for the considerations of this work. The terms *process* and *business process* will be used synonymously throughout this work. In contrast to the above definition this work distinguishes between the *description* of activities and their relations and the actual *execution* of tasks. The description will be referred to as the *process*, while the executable instance of the process will be referred to as the *case*. This differentiation is rooted in the theory of case handling as introduced by van der Aalst and Berens in [122] and comprehensively discussed in [96; 105; 125]. Executing a case temporarily requires resources like material, workforce, or capital. These resources can either be consumed by the case or used and conserved for further cases. Russel et al. give a detailed discussion of the properties of human and non-human resources in [100]. Since various approaches to assess business processes are applied by organizations [8; 34] different methods, modeling languages, and tools were developed.

A business process is usually defined in subsequent steps. The *process documentation* aims at describing the current state of the business as a base for further formalization. To document processes a low level of detail is required so that tables or textual descriptions are the method of choice. Such documentations help to establish process oriented thinking and give the employees a holistic overview and basic understanding of the business functions and the interfaces between business departments.

Based on process documentations, *formal process models* can be developed on different levels of abstraction. The abstraction levels may be connected



by the nesting of model parts (i.e., subprocesses). For different purposes, different modeling languages are available. Examples of modeling languages for business processes are the Business Process Modeling Notation (BPMN, [46]), UML Activity Diagrams (UML AD, [45]), Event-driven Process Chains (EPC, [60]), Petri Nets (PN, [92]), and Yet Another Workflow Language (YAWL, [123]), just to mention a few.

The *identification of responsibilities* for the single working steps uncovers the enterprise's distribution of competences and may help to systematically reduce the needed communication efforts.

Formalized process models allow for the *definition of key process indicators (KPIs)*. KPIs can be defined for amount-, time-, and cost-related dimensions. Evaluating measurements allows for the *controlling of processes* and may motivate business process reengineering (BPR) projects [56]. A further purpose of formalized process models is to implement *rules of action* on the operational level. These may—annotated with target values—enable the conclusion of *service level agreements*.

### 1.1.2 Mobile Business Processes

Business processes in distributed and/or volatile business environments are increasingly gaining attention regarding efforts to improve the efficiency of process execution. Besides the constant pressure to improve their performance by means of change that all organizations experience [12], the main technical reasons are increasing availability of high-bandwidth data communication networks [58] and rapid development of mobile devices with ergonomic graphical user interfaces.

Process related data can now be processed both downstream (i.e., toward the mobile worker) and upstream (i.e., toward the organization's information systems). This allows for mobile business processes to become controlled centrally by utilizing a mobile workforce management system [49] (see Section 1.1.3).

Mobile processes are characterized by spatially distributed resources requiring work to be performed at their respective locations. Thus, the performing workforce must be mobile. Among others, such resources can be technical equipment or customer sites. Consequently, mobile processes are processes depending on utilizing externally located resources. Extending the process definition given by van der Aalst and van Hee in [124], in this work mobile processes are referred to as business processes of which at least one activity takes place outside the organization's physical bounds—i.e., in the field [47; 77]. Concerning such processes, we will later distinguish between *simple mobile processes*—consisting of just one mobile activity—and *complex*

*mobile processes*—consisting of more than one mobile activity. The reasons for this distinction are the different control and scheduling requirements of both types of processes. In opposition to simple mobile processes, complex mobile processes require temporal and spatial synchronization of workers at different locations.

Examples of mobile processes are the maintenance of supplier networks (electrical power, gas, water, etc.), the service of technical equipment at customer's sites, sales, and the monitoring of distributed sites as construction sites [78] or supermarkets (e.g., mystery shopping [131]).

Mobile processes possess a number of attributes distinguishing them from non-mobile processes. The most important one is the spatial distribution of activities because they are bound to external locations. The resulting spatial distribution of mobile processes represents a challenging restriction for scheduling the workforce, which is the assignment of resources to mobile activities [72], in comparison to scheduling in non-mobile processes [100]. It forces:

- workers and equipment to move to the site where the activity is executed,
- collecting and processing information about the location of resources (workforce, equipment, material),
- communication over wireless networks outside of the organization's sphere of influence; and, thus, possible data and voice communication cut-offs during process execution,
- possibly delayed updates of execution states of mobile activities in central information systems,
- preparing work lists for the workforce and preloading data to the workers' mobile devices, and
- considering the spatial distribution of skills and competence over time.

The spatial distribution of skills and competence must be considered especially if highly prioritized cases requiring immediate execution can occur. Creating work lists is the direct consequence of enormous amounts of non-productive times for transport and travel. With work lists activities can be ordered such that transport efforts become reduced. Further reasons for creating work lists are the availability of capacity constrained resources as special equipment and the prioritization of certain processes or activities. Thus, it is necessary to apply workforce scheduling methods in mobile process environments.

In summary, the mobile processes considered in this work are based on the definition given by van der Aalst and van Hee (see p. 2) and are therefore defined as follows:

A mobile process consists of a number of tasks that need to be carried out outside of the physical bounds of the enterprise, and of a set of conditions that determine the order of the tasks. A task is a logical unit of work that is carried out as a single whole by a worker or a group of workers. The workers are controlled by means of a workforce management system. Tasks can be interrupted by the workforce management system causing the workers to perform different tasks of higher importance.

Enterprises that cover a certain size of area and run mobile processes tend to be organized in subdivisions of which each is responsible for regionally assigned assets and the respective business processes. Thus, the enterprise administers a number of regional divisions with identical functional processes performed on their respectively distributed assets. Such assets can be dedicated predefined customer sites, immobile technical equipment, or a variety of different consumer products at unforeseeable sites—if the company offers appropriate repair services, for instance. The definition of regions is often based on the historical evolution of the enterprise, on administrative structures as counties or states, or on natural boundaries like rivers or mountain ranges. For an organization its current assignment of assets, resources, and workers to regions may have sensible reasons at first glance, but the organization can hardly determine if the assignment is optimal regarding the organization's cost situation. This aggravates on the occasion of mergers or acquisitions, since there is usually no prior knowledge about the performance of the unified workforce. Cost functions may be based on travel distances, duration of technical malfunctions, or the number of workers employed. They strongly vary between organizations with respect to their different areas of business and business goals.

### 1.1.3 Workflow Management of Mobile Business Processes

A workflow can be seen as the part of a business process that is qualified to be performed automated or supported by information technology. The subject matter of *workflow management systems* is the technological support and control of business processes. They are information systems used to plan, support, and automate business processes and to control the operations between different resources (humans and / or applications) by seamlessly integrating heterogenous, widely distributed data sources and information systems [124]. Process data is being routed between human process

participants and applications. The underlying process definitions are stored separately from the actual control software as part of the system's configuration. Hence process change—caused by, e.g., strategic business goals or operational constraints—is not adapted by modifying the control software but by reconfiguring it. Workflow management systems are utilized to control and automate both industrial manufacturing (e.g., production workflow) and bureaucratic handling (e.g., finance, insurance, public services) [121]. Implementing a workflow management system allows for restructuring processes with various optimization objectives.

The efficiency of workflow management systems is determined fundamentally by the stability of the concerned business processes over time. Workflow management systems may help to considerably reduce throughput times and handling times of cases, format mismatches, and erroneous handling paths. Furthermore, they increase the quality of processing and up-to-dateness of reporting as long as process definitions and organizational structures alternate less than anticipated at design time. This stability is particularly important regarding the execution of different cases of one and the same process.

Another duty of workflow management systems is to achieve format continuity between the various process participants and involved information systems. An essential contribution to format continuity is the absence of paper based forms from the concerned processes. To achieve this optimization goal it is essential to connect the involved resources by means of a computer network [124]. Since only recently reliable high-bandwidth wireless data communication networks became available, permanent network connections could not be provided in the past. Thus, supporting mobile processes by workflow management systems required either immense organizational effort via cellular phones or state information of processes was not available in real-time.

Due to the increasing availability of cellular wireless data networks, mobile businesses gain manifold possibilities for electronically supporting and controlling their mobile business processes. Introducing workflow management systems into such processes can be achieved by equipping the mobile workforce with mobile electronic devices capable of communicating with the enterprise's network wirelessly. Utilized in mobile environments, workflow management systems face increased requirements concerning their functionality.

Because of their administrative characteristics, workflow management systems for mobile processes are also referred to as workforce management systems. We will use both terms synonymously throughout this work. Compared to non-mobile workflow management systems, their additional features are workforce scheduling based on individual qualifications and geographic location as well as the bi-directional supply of activity-based and location-based

data in real-time. Utilizing workforce management systems in mobile environments under the circumstances mentioned above implies enormous challenges concerning their implementation and underlying process definitions. Reasons for this are:

**Technological:** Neither *bandwidth* nor general *availability* of current wireless data networks are predictable with certainty. *Capabilities of mobile devices* differ both compared to stationary devices and among each other. Mobile devices often face restrictions (e.g., reduced display size, user interface, computational power) not applying to stationary devices. Besides restrictions of devices' capabilities, applying them in mobile processes generates additional requirements to *ergonomics* (e.g., single-hand control, readability under exposure to direct sunlight, amount of data to be input).

**Processual:** During the execution of mobile processes situations may occur that demand workers to interrupt their current work or to postpone currently planned work to execute activities not planned yet. This may be due to the occurrence of emergency situations located closely to a worker's current location, forcing this worker to react immediately. Such situations are not restricted to mobile processes, but in mobile environments the impact on the business is larger than in non-mobile processes. Thus, the a-priori definition of working scenarios may become challenging, if not impossible.

**Legal / Societal:** In numerous countries mobile workers are protected from real-time monitoring of their geographic location by legal regulations. In addition, the workers themselves may dislike such monitoring. If the workforce's retentions are not taken into account, their motivation may decrease. Nonetheless, it is necessary to gain such information to achieve flexible operative control of their working plans in emergency situations.

All these reasons imply that workforce management systems in mobile environments must be able to react more flexibly to the complex functional requirements of their environment as well as to fulfill harder non-functional requirements than systems in non-mobile environments. The non-functional requirements may be so restrictive that functional requirements are influenced and are thus subject to temporary or even permanent change. If, for instance, it is prohibited by law to acquire the exact geographical location of a worker by using the Global Positioning System (GPS) or similar technologies, as a result guaranteed emergency reaction times may not be met and service level agreements of the business must be adjusted accordingly. In this example imprecise information about the worker's current position can

be gained from the location of his activities and the time intervals between state updates unless the worker could not send state updates due to data communication cut-offs. It can be assumed that the problems arising from technological reasons will be compensated by technological progress in the future while this is not foreseeable for the processual and legal reasons.

In this work we assume that the mobile workers spend most time of their working duty in the field, that the activities of one and the same worker can belong to different business cases and to different business processes, and that the mobile processes are supported by workforce management systems.

#### 1.1.4 Performing Business Change

By defining business processes, organizations empower themselves to assess the qualitative properties of their business operation [8]. Though this gives insights into the structure of business processes, improving the business operation and predicting necessary resources require to assess quantitative properties. BPR projects consist of at least two stages: (i) the analysis of the business' current state and the development of a concept for changes, as well as (ii) the actual realization of the changes to the business (processes). In the conceptual stage business analysts face the necessity to predict the practical and economic outcomes of the project's realization. Such predictions may be based on static properties of the business processes as, for instance, the expenditure of time for certain activities, the number of activities a worker can perform in a given period of time, or the amount of energy consumed by a business case. The necessary information can be gained by observing and measuring the actual process execution, given that appropriate attributes to be measured are defined, sensors are implemented, and the processes are run repeatedly. Nonetheless, highly dynamic or complex systems, processes, and environments need to be examined regarding their respective dynamic properties (e.g., working-waiting-ratio of the workforce, number of workers available at given intervals, utilization of resources) during while estimating the project's outcomes. Since many business processes are too complex for mathematical analysis [62] or their execution is too dependent from dynamic properties [49], process analysis is best performed by stepwise simulation [116]. Simulation is a well established method to gain insight into the dynamic properties of such environments [39]. Simulations can bring up results identifying resource load and possible bottlenecks. Such results are utilized to create, evaluate, adjust, and discard hypotheses for the iterative improvement of process models until a process set to be implemented is found. Figure 1.1 illustrates the procedure.

Since an organization usually runs different processes, it is likely that business analysts develop numerous process models of varying business scope

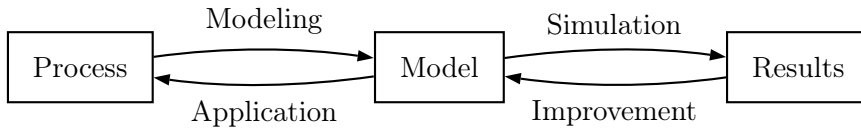


Figure 1.1: Simulation of process models (following [98])

during BPR projects. It is obvious that these models abstract from domain specific properties of mobile work as, e.g., workers' schedules, travel expenses, wireless data connections, but cover the operative properties of the processes. Nonetheless, domain properties are the main drivers of the process examination objectives. These objectives are bound to the answers to the following questions:

What is

- the best geographical distribution of workers and skills,
- the best suited scheduling algorithm, and
- the best assignment of equipment to workers?

This list of items is by far not complete, but is meant to give an idea of questions process improvement projects face. When considering a solution to be *the best* for one of these questions, we always mean that this solution should induce the lowest total cost of operation. Due to the dynamic nature of the domain properties, these questions may not be solved by static analysis but by simulating the situation. A simulation method for mobile businesses must thus be capable of introducing the dynamic properties into the simulation while preserving the functional properties of the process models.

Although business process simulation (BPS) is widely accepted in research communities, an empirical study [79] discovered, that 64% of the enterprises examined do not use simulation to predict the outcomes of change projects in advance. This may be due to the broad focus of simulation tools which aim to cover most of the problem domains suitable for simulation. In [24] a BPS template for the Arena<sup>1</sup> simulation suite is presented to support appropriate industrial projects. Though BPS is increasingly gaining attention, most approaches focus on single business processes, but only marginally consider the properties of the process' environment—where several concurrent processes are executed.

Enterprise Modeling (EM) aims to overcome the aforesaid limitations of BPS approaches by considering the organization as a whole with its complete set

<sup>1</sup>see <http://www.arenasimulation.com>

of business processes. Enterprise models facilitate the model-driven design, analysis, and operation of an enterprise [33]. Many early EM approaches (e.g. [26; 68; 130]) mainly focus on either enterprise models for production and manufacturing environments or on documenting and understanding organizational structures to develop Enterprise Resource Planning (ERP) systems. They are not intended to be domain-independent frameworks to integrate variable BPR models. In [9] the Common Enterprise Modeling Framework for Distributed Organizations (CEMF-DO) is proposed. This framework aims at utilizing the Unified Enterprise Modeling Language (UEML) [126] to gain interoperability of different distributed enterprise models.

## 1.2 Identified Problem and Contribution of this Work

### 1.2.1 Modeling and Simulation of Mobile Business Processes

Business process modeling aiming to simulate the processes requires to model all aspects of the processes and the environment they are executed in. Otherwise the simulation results are not comparable to real-world data. In most cases businesses optimize their processes with respect to their function. The optimization potentials left over are usually not bound to what the workers do and how they do it. Further optimization needs to consider how the process execution is organized. Modeling process aspects not directly related to the business purpose of the processes generates extra work, especially if the number of non-functional aspects is large and these aspects are relevant to most of the processes to be modeled. This is the case with mobility and mobile business processes. Mobile business processes are influenced by a large number of properties which are independent from the business purpose of the processes. Examples for such properties are the traveling of workers, the synchronization of tasks, and fluctuating data connections. There should be no need for the average business user, who just wants to model a specific business process, to model the constant properties and restrictions of mobility explicitly and repeatedly. Not only the number but the complexity and interdependence of these properties leads to modeling situations hard to handle, since only few adequate modeling languages exist. These modeling languages (e.g., colored Petri nets—CPN) tend to demand too much from the average business analyst due to their expressive power gained from a very small set of notation elements. Small sets of notation elements increase the size of models even for simple business processes, which again increases the modeling effort. Under such circumstances the usual solution is to apply abstract, business independent domain models. Currently no domain model of mobility capable of simulating various independent business processes simul-



## 1.2. IDENTIFIED PROBLEM AND CONTRIBUTION OF THIS WORK<sup>11</sup>

taneously is available to the public. This work introduces a general domain model of mobility in Chapter 2. The general domain model is the basis for a model capable of simulating independent business processes as introduced in Chapter 3.

Additionally, this work aims for the radical reduction of modeling complexity by separating the modeling of mobility properties from the modeling of process properties. This approach virtually splits the modeling performed in BPR projects into two stages [50]:

1. A highly specialized and complex domain model of mobility is developed based on CPNs. The model covers the properties and behaviours of mobility and can be utilized in BPR projects. This work provides such a reusable and simulatable domain model of mobility in Section 3.2.
2. Process models are developed by business analysts in a—compared to CPNs—more common language based on UML activity diagrams. This Simple Mobile Process Language (SMPL) and the resulting process models consist of only very basic control structures. The SMPL process models can be transformed to the notation of the CPN domain model of Section 3.2 and then be executed in a simulator. Sections 3.3 and 3.4 of this work describe the SMPL and the transformation of modeled processes to simulatable models.

In Chapter 5, a set of software tools supporting the analysis of mobile business processes by means of the methods mentioned above is presented.

### 1.2.2 Workforce Scheduling

To management individual work lists, planning the workforce utilization is of particular importance in mobile environments. Creating work lists—so called schedules—for the workers is related to routing problems in logistics, like the Vehicle Routing Problem (VRP) [115], as well as to resource scheduling problems in production environments, like the Resource Constrained Project Scheduling Problem (RCPSP) [67]. Current scheduling methods for the mobile workforce are only known for environments where the processes consist of just one activity. It was already outlined above that the processes considered in this work may consist of more than one activity and that these activities may depend on each other by demanding parallel or subsequent execution. Since the contribution of this work, as outlined in the preceding section, requires to simulate mobile processes and thus to create work lists, the lack of appropriate scheduling methods has to be addressed.

Thus, this work will present a formulation of the Mobile Workforce Scheduling Problem with Multitask-Processes (MWSP-MP) in Chapter 4. Furthermore, a set of neighborhood operators on schedules necessary for the application of meta-heuristics to the MWSP-MP, as well as methods for the generation of start solutions of the MWSP-MP are presented.

### 1.2.3 Solution Overview

The activity diagram given in Figure 1.2 illustrates how business analysts performing BPR projects can benefit from the application of our findings. The remarks of the different activities refer to the supporting elements developed in this work.

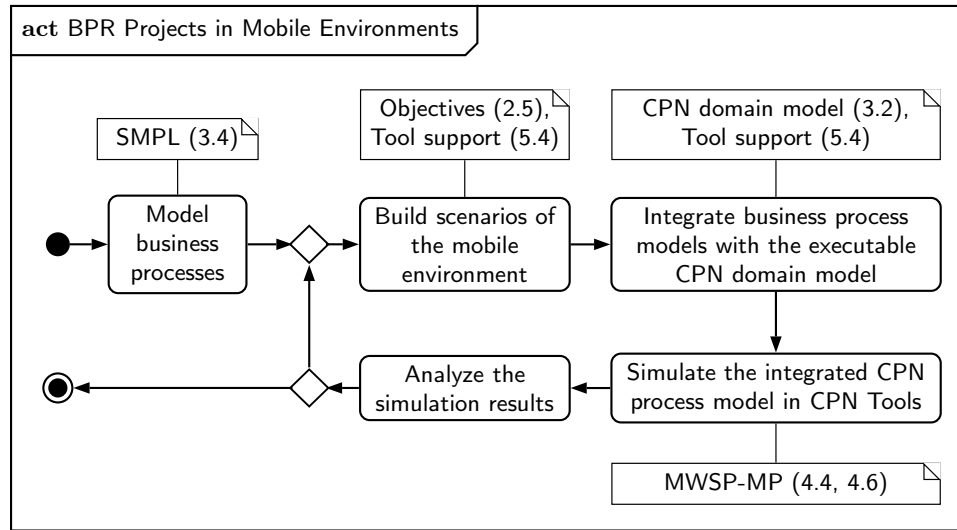


Figure 1.2: Course of BPR projects in mobile environments as seen by this work

## Chapter 2

# Domain Model of Mobility

The mobile business processes considered in this work run in mobile environments with common properties that are independent of the business domain. It is thus efficient to provide a process modeling approach hiding the common properties of mobility from business process modelers and supporting to focus on the actual business dependent properties of the processes. The common properties of processes resulting from mobility should be propagated to the process models in an automated way. This chapter introduces a business independent model of mobile work to be utilized to provide an accordant modeling environment. The model is not only useful for the purpose of this work but may also act as a reference model of mobile work to appeal a broader audience.

In the first section examples of mobile processes are described. The following sections introduce an abstract process model, entities of mobile environments, state models of mobile work, and optimization objectives applied to BPR in the context of mobility.

### 2.1 Utility Industry Process Examples

Throughout this work mobility aspects will be explained and discussed with examples of mobile business processes from the utility domain. Therefore, a few typical processes will be introduced here. All processes and data presented in this work originate from several business process consulting projects the author accomplished with a German power and gas utility in the years between 2004 and 2010.

Even though the examples originate from the utility industry, similar processes can be found in other industries, too. Examples for such industries are telecommunications, logistics, civil service, and disaster operation.

### 2.1.1 Asset Inspection

Inspections of technical equipment in the field as, e.g., power substations or gas pipelines are usually performed periodically and are thus a major part of the operation of a utility. Inspections do not intervene in the operation of the networks, but provide information about the technical state of the equipment. Thus, an inspection is a simple business case, involving just traveling to the location of the asset and performing the inspection itself. Inspections are considered simple mobile processes.

**Definition 2.1 (Simple mobile process)** *A mobile process is considered simple if it consists of exactly one mobile activity. Accordingly, a mobile case defined by a simple mobile process consists of exactly one mobile task. A worker executing such a task just has to drive to the location of the task, execute, and finalize the task. The execution of a case in the field is finished with finishing its single task.*

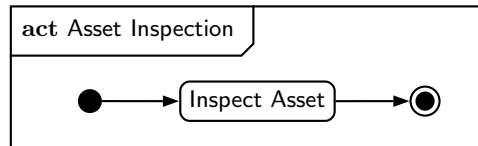


Figure 2.1: Asset inspection

Examples for further simple processes in the utility industry are to read power meters at connection points, or to encash fees owing. Figure 2.1 shows the activity diagram of an inspection.

### 2.1.2 Switched Power Line Processes

Figure 2.2 shows a typical situation after a new power substation was erected and needs to be connected to the existing power network. In a substation medium voltage of about 20 kV is transformed to low voltage of about 400 V. It enables a utility to deliver power on the last mile to the customer. The newly erected substation is connected to the existing network by inserting a sleeve (tee branch) at location L3 into the power line between the stations at locations L1 and L2.

Safety concerns demand that the power line between the stations at L1 and L2 has to be turned off before the sleeve is inserted and turned on again after the work is done. Fig. 2.3 shows the resulting process as an activity diagram.

As long as the line is turned off no energy is sold to customers connected to that respective power line between L1 and L2. Thus, it is desirable to minimize the downtime of the line. As a direct consequence, different workers

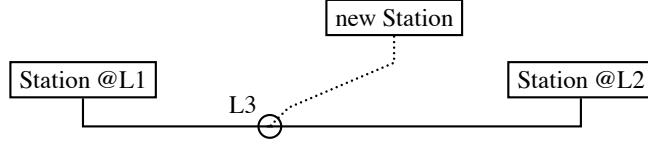


Figure 2.2: Power line extension situation

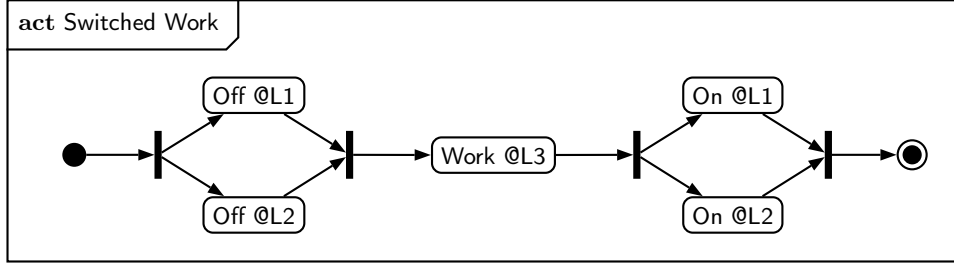


Figure 2.3: Power line extension process

should perform the tasks at the different locations to avoid downtimes of the line resulting from the travel effort a single worker would face. For our example, this means that different workers may turn the stations on and off while a third worker enters the site L3 to insert the sleeve. The five mobile tasks depicted in Figure 2.3 are closely coupled in time, while possibly far apart in space. As a result actually independent processes become interdependent due to the fact that one worker can be involved in several different processes while maybe performing just one activity per process. In this way, delays that occur at a certain site may cause massive delays at completely different sites and processes. Switched power line processes are considered complex mobile processes.

**Definition 2.2 (Complex mobile process)** *A mobile process is considered complex if it consists of more than one mobile activity. Accordingly, a mobile case defined by a complex mobile process consists of more than one mobile task. Workers executing tasks of complex mobile cases have to consider precedence constraints of the different tasks and, thus, must synchronize their work. The execution of a case in the field is finished with finishing its last task.*

### 2.1.3 Damage Search and Repair

Figure 2.4 shows a typical situation after a power line was damaged due to an unknown reason. The location of the damage may be estimated to be somewhere between L1 and L2 by the central network control unit of the utility, but the exact location has to be detected by field workers. In this

particular case we assume that the line between the stations at L1 and L2 consists of a subterranean part (also called *cable*) and an elevated part (also called *wire*). Such a setup can occur if, e.g., a power line crosses a river.

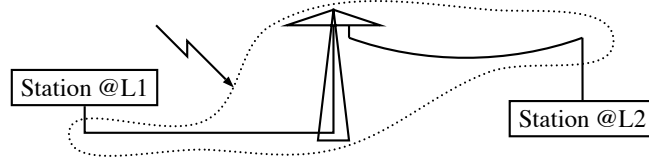


Figure 2.4: Power line damage search and repair situation

The resulting process (see Figure 2.5) consists of a common search branch and different repair branches based on the type of the damaged part of the power line. This is due to the fact that a buried cable has to be excavated by construction workers before electricians can repair the damage while for repairing a broken wire no external support is needed. Both branches after the decision node in Figure 2.5 may further consist of power line switching activities (see Figure 2.3), which is not necessary to be considered for the purpose of this work.

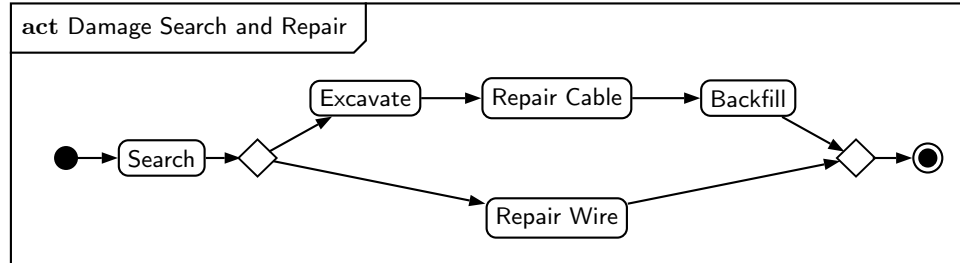


Figure 2.5: Power line damage search and repair process

As for the switched powerline process depicted in Figure 2.3, it is desirable to reduce the downtime of the damaged line. Since damages occur unplanned during the network's operation, scheduling the workforce has to be executed in a preemptive manner—leading to interrupts of currently running cases based on process priorities, total costs of downtimes, travel effort, and workforce utilization. As a result, the involved processes are also bound to interdependencies not founded in the business case, but in the properties of the business domain “Maintenance of power networks”. The damage search and repair process is a complex mobile process as well.

## 2.2 Abstract Model of Mobile Processes

Based on the definition of mobile processes given in Section 1.1.2, this section offers deeper insight into the properties of mobile processes as considered in this work. To develop a business independent domain specific model of mobility as well as of the execution of mobile work, it is necessary to understand the differences and commonalities of mobile business processes. Figure 2.6 gives an overview of the typical administrative parts of any mobile process:

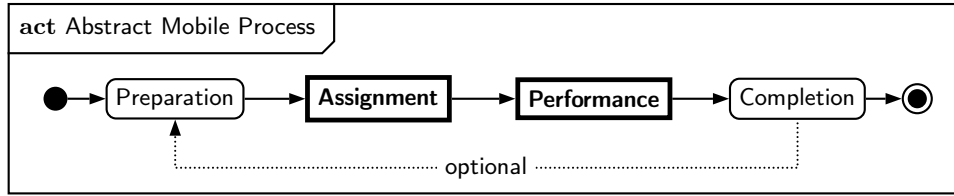


Figure 2.6: Abstract mobile process

**Preparation** of a mobile task includes gathering and bundling necessary information and material. It is carried out at the organization's headquarter or at the depot of the region. An example for preparing the inspection of a municipality's power network is to print both the according network's map and the listing of all points of special interest as, e.g., power substations.

**Assignment** of a mobile task to a worker appoints the worker to execute this task. For the assignment it is necessary to determine the appropriate qualification of workers capable to execute the task and to determine the scheduling algorithm. To assign a task to a worker, the task's required qualifications must be in the set of the worker's skills. Due to the fact, that traveling contributes considerably to the costs of mobile processes, it is not possible to provide the worker with work items during the working day, but it is necessary to manage work lists which contain a worker's activities for a given period of time (e.g., one day) in chronological order. The assignment of the task to the worker is performed manually or automatically and is carried out at the organization's headquarter or at the depot of the region. Data may be transferred to the worker at the headquarter (face-to-face) or via wireless data communication.

**Performance** of a mobile task includes that the worker travels to the site of work, the actual work, and that the worker gathers work-related information on-site. It is mainly carried out at the task's location.

**Completion** of a mobile task includes analyzing the work results, accounting, updating technical data, and planning further measures, if neces-

sary. It is typically executed at the headquarter or at the depot of a region of the respective organization.

As marked in Figure 2.6, the steps considered in this work are the assignment of mobile tasks to workers and the performance of the mobile tasks. These are the process steps that are influenced by the properties of mobility.

### 2.2.1 Assignment of Mobile Work

The primary goal of assigning mobile work is to minimize the total process costs of an organization. An in-depth discussion of these costs and their relations is presented in Section 2.5. The resulting secondary goals of assigning work are to execute the cases with the highest priority first, to reduce travel effort, and to avoid workers having idle times. The general assignment activities are depicted in Figure 2.7.

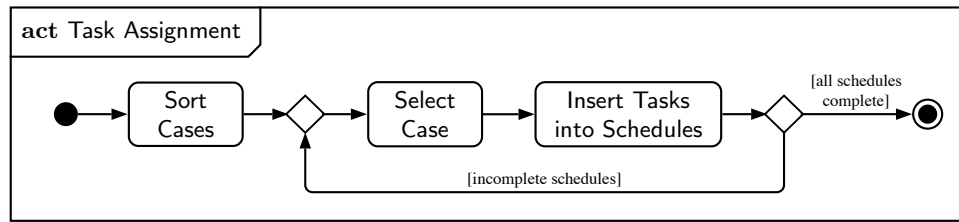


Figure 2.7: Assignment of a mobile task

The assignment can either be performed manually or automatically. If it is performed manually, a foreman or dispatcher allocates the work items to the workers based on the experience as well as on the spatial distribution of the workers. If, alternatively, the assignment is performed automatically by a workforce management system, the system allocates the work items based on preset priorities, expected travel times, and interdependencies of the cases and tasks. Nonetheless, for the general course of the assignment, it is not necessary to differentiate between manual and automatic assignment.

At the beginning of a planning period the pending cases are sorted by priority. The resulting list of the cases is then processed top-down. Every case is split into its tasks, and the tasks are inserted into the schedules of suited workers. Whether a worker is suited for executing a certain task depends on his skills, on his geographic location, and on the tasks already present in his schedule. Algorithms for solving the respective workforce scheduling problem will be presented and discussed in Chapter 4. After a case is inserted into the workers' schedules either all schedules are complete or not. The assignment of mobile work is finished as soon as all schedules are complete, otherwise the next case is selected and inserted into the schedules still



incomplete. A schedule is considered complete if it contains tasks for a whole planning period—e.g., expected working and travel times sum up to eight or nine hours of a working day.

### 2.2.2 Performance of Mobile Work

Independently from the business objectives, the performance of a mobile business process follows a general scheme. For each task of a process the steps depicted in Figure 2.8 have to be executed.

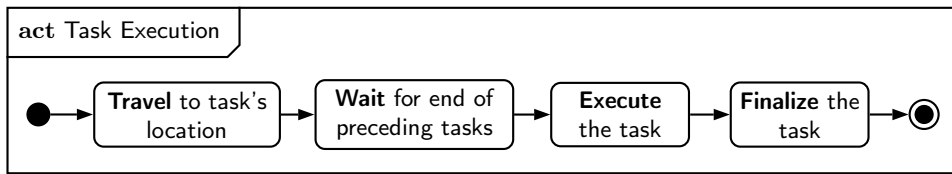


Figure 2.8: Execution of a mobile task

**Traveling** is the transfer of the worker responsible for executing a task to the location of that task. It starts at the depot of this worker (for the first task of the day) or at the location of another task that has been accomplished by the worker before. In this work, traveling *to* a task's location is considered to belong to that task. In contrast traveling *from* the task's location does not belong to that task but to the next task in the worker's schedule.

**Waiting** may be necessary after the location of the task has been reached by the worker and either (i) the time window of the task has not been reached yet or (ii) a preceding task of the case has not been finished yet. As an example for the latter consider the process depicted in Figure 2.3. The task “Work @L3” must not start until the tasks “Off @L1” and “Off @L2” are finished.

**Execution** of a task refers to accomplishing the business objective of this part of the case. In this work, the execution of tasks is considered from an administrative point of view only. It is thus considered as an economically weighted period of time which blocks a worker at a certain location with certain costs. The actual business value added does not need to be considered for the purpose of this work.

**Finalization** of a task involves gathering data that describes the work results. This includes business data as, e.g., the nature and amount of material consumed as well as administrative data as, e.g., consumed time.

## 2.3 Entities of Mobile Work

In the following the entities involved in the execution of mobile processes will be introduced and defined formally. Even though most of the entities defined in the following will be referenced later in this work, it is also intended to formulate a reference set of entities of mobile work. Since several different definitions exist for the terms used, this work mostly follows the textual descriptions given by van der Aalst and van Hee in [124].

**Definition 2.3 (Location)** *A location is a geographical position on the surface of the Earth. It is determined by its longitude and latitude. A location  $l$  is denoted by the tuple  $l = (l^{lon}, l^{lat}) | l^{lon} \in [-180.0, 180.0], l^{lat} \in [-90.0, 90.0]$  with  $l^{lon}$  being the longitude and  $l^{lat}$  being the latitude of  $l$ .*

**Definition 2.4 (Qualification)** *A qualification constitutes if a worker has the necessary education or experience to perform a certain task. For a task a qualification  $q \in \{0, 1\}$  denotes whether the task demands this qualification ( $q = 1$ ) or not ( $q = 0$ ). For a worker it denotes whether the worker has this qualification ( $q = 1$ ) or not ( $q = 0$ ). The set of all qualifications of an organization is denoted by  $\mathcal{Q}$ .*

An example of a qualification is “Authorized to handle conducting electrical equipment”.

**Definition 2.5 (Activity)** *An activity describes a generic unit of work. It is not bound to the execution of the work it describes. An activity can not be split in pieces, it is thus atomic. For an activity  $a$  the default execution time  $t_a^{def}$  and the required qualifications  $q_a^{req}$  are declared. The set of all activities of an organization is denoted by  $\mathcal{A}$ . An activity  $a \in \mathcal{A}$  is denoted by the tuple  $a = (t_a^{def}, q_a^{req}) | t_a^{def} \in \mathbb{N}, q_a^{req} = (q_1, \dots, q_n), q_i \in \{0, 1\}, n = |\mathcal{Q}|$ . Any  $q_i = 1$  indicates that activity  $a$  requires the performing worker to have qualification  $q_i$ .*

An example for an activity is “Turn on a power substation”.

**Definition 2.6 (Task)** *A task is an instance of an activity  $a \in \mathcal{A}$ . It is defined by the properties of the activity it instantiates, and, additionally, by a time window as well as by a location. The time window determines the period for the execution of the task. A task must not start before the start of its time window, and a task must not end after the end of its time window. The location denotes the geographic position at which the task has to be executed. A task is defined as a tuple  $\tau = (a, [t_\tau^{min}, t_\tau^{max}], l) | a \in \mathcal{A}, [t_\tau^{min}, t_\tau^{max}]$  being the time window of  $\tau, l$  being the location of  $\tau$ . The set of all tasks of an organization is denoted by  $\mathcal{T}$ .*

An example for a task is “Turn on power substation U5314”.

A business process is an ordered set of activities describing the way a certain type of product or service can be produced or delivered. A mobile process is a business process that involves at least one activity taking place outside the physical bound of the organization—i.e., in the field. A mobile process is composed of activities, control structures, and directed edges denoting the direction of the control flow. Control structures define the rules of process execution. Different paths in a mobile process may be executed either parallel (all paths are executed) or exclusively (exactly one path is executed). The set of the control structures is  $\mathcal{CS} = \{Start, End, AND_{split}, AND_{join}, XOR_{split}, XOR_{join}\}$ . Since one and the same activity or control structure may appear several times in a process, they must be uniquely identifiable. This is achieved by introducing process nodes. A process node is a tuple  $(id, as) \mid id \in \mathbb{N}, as \in (\mathcal{A} \cup \mathcal{CS})$ .

**Definition 2.7 (Mobile Process)** *A mobile process is a directed graph consisting of nodes connected by edges. It is defined as a tuple  $p = (\mathcal{N}_p, F) \mid \mathcal{N}_p = \{(id, as)\}, id \in \mathbb{N}, as \in (\mathcal{A} \cup \mathcal{CS}), F \subseteq (\mathcal{N}_p \times \mathcal{N}_p)$  where  $F$  defines the edges between activities and control structures. The set of all mobile processes of an organization is denoted by  $\mathcal{P}$ .*

An example for a process is “Replacement of a power substation”.

**Definition 2.8 (Case)** *A case is an instance of a process  $p \in \mathcal{P}$ . It describes the production or delivery of a concrete product or service. A case has a finite lifetime. It is defined as a tuple  $c = (\mathcal{T}_c, [t_c^{min}, t_c^{max}]) \mid \mathcal{T}_c \subseteq \mathcal{T}$ .  $\mathcal{T}_c$  is the set of all tasks belonging to case  $c$  and  $[t_c^{min}, t_c^{max}]$  is the time window for the completion of  $c$ . The set of all cases existing in a system under consideration is denoted by  $\mathcal{C}$ .*

An example for a case is “Replacement of power substation U5314”.

**Definition 2.9 (Task’s timestamps)** *For all tasks  $\tau \in \mathcal{T}$  three timestamps are defined.  $t_\tau^{arrive} \in \mathbb{N}^+$  denotes the timestamp at which the worker performing this task arrives at the location of  $\tau$ .  $t_\tau^{start} \in \mathbb{N}^+$  denotes the timestamp at which the worker performing this task actually begins to perform  $\tau$ .  $t_\tau^{finish} \in \mathbb{N}^+$  denotes the timestamp at which the worker performing this task finishes the work on  $\tau$ .*

**Definition 2.10 (Case’s timestamps)** *For all cases  $c \in \mathcal{C}$  two timestamps are defined. For a case  $c \in \mathcal{C}$ ,  $t_c^{start} \in \mathbb{N}^+$  denotes the timestamp at which the first task  $\tau_f \in \mathcal{T}_c$  is started ( $t_{\tau_f}^{start}$ ), and  $t_c^{finish} \in \mathbb{N}^+$  denotes the timestamp at which the last task  $\tau_l \in \mathcal{T}_c$  is finished ( $t_{\tau_l}^{finish}$ ).*

**Definition 2.11 (Worker)** A worker is a mobile resource that performs tasks at locations. A worker is denoted by a tuple  $w = (q_w^{avail}, l_w, l_w^{depot}) | q_w^{avail} = (q_1, \dots, q_n), q_i \in \{0, 1\}, n = |\mathcal{Q}|, l_w$  being the current location of  $w$ , and  $l_w^{depot}$  being the depot where the worker starts and finishes his planning period. The qualifications of  $w$  are denoted by  $q_w^{avail}$ . Any  $q_i = 1$  indicates that worker  $w$  has qualification  $q_i$ . The set of all workers of an organization is denoted by  $\mathcal{W}$ .

**Definition 2.12 (Asset)** An asset is a stationary resource or piece of equipment. An asset is denoted by a tuple  $e = (id, l) | id \in \mathbb{N}$  being a unique identifier of the asset and  $l$  being the location of the asset. The set of all assets of an organization is denoted by  $\mathcal{E}$ .

An example for an asset is “power substation U5314”.

**Definition 2.13 (Region)** A region is a part of the area covered by an organization. It is determined by its assigned workers, assets, and the frequency of process execution. The set of all regions is denoted by  $\mathcal{R} = \{r_1, \dots, r_n\}$ . A region  $r \in \mathcal{R}$  is then defined as a tuple  $r = (\mathcal{W}_r, \mathcal{E}_r, \mathcal{I}_r, l_r^{depot}) | \mathcal{W}_r \subseteq \mathcal{W}, \mathcal{E}_r \subseteq \mathcal{E}, \mathcal{I}_r \subseteq (\mathcal{P} \times \mathbb{Q})$ .  $\mathcal{I}_r$  is a set of tuples denoting the frequencies (or incidences) of process execution for each process being performed in the region  $r$ . Given a tuple  $(p, f) \in \mathcal{I}_r$ ,  $f \in \mathbb{Q}$  denotes the average number of executions of  $p \in \mathcal{P}$  during one planning period.  $l_r^{depot}$  denotes the location of the depot of  $r$ . It is further necessary that each worker and each asset belong to exactly one region:  $\mathcal{W}_i \cap \mathcal{W}_j = \emptyset \wedge \mathcal{E}_i \cap \mathcal{E}_j = \emptyset \wedge \bigcup \mathcal{W}_i = \mathcal{W} \wedge \bigcup \mathcal{E}_i = \mathcal{E} \forall i, j \in \{1, \dots, n\}$ .

**Definition 2.14 (Planning Period)** A planning period is a time span during which business operations are typically performed—i.e., cases and tasks are executed by mobile workers. It is defined by its beginning time  $t_{period}^{start}$  and its finish time  $t_{period}^{finish}$ .

Very often the planning period can be seen as a working day, starting in the morning and ending in the evening. Nonetheless, planning periods can be any time interval. The finish time is usually just an administrative convention, since unexpected situations during the business operations may extend the actual work duration beyond this time.

**Definition 2.15 (Schedule)** A schedule is the daily worklist of a worker. The schedule  $\sigma_w$  of worker  $w \in \mathcal{W}$  is a tuple  $\sigma_w = (\tau_1, \dots, \tau_n, \tau_{depot}) | \tau_i \in \mathcal{T}, n \in \mathbb{N}$ .  $\tau_{depot}$  denotes a virtual task with the duration 0 situated at the depot of the worker  $w$ . For any task  $\tau$  and any schedule  $\sigma$  the function  $elem_\sigma(\tau)$  returns true if  $\tau$  belongs to  $\sigma$  and returns false, otherwise. For any schedule

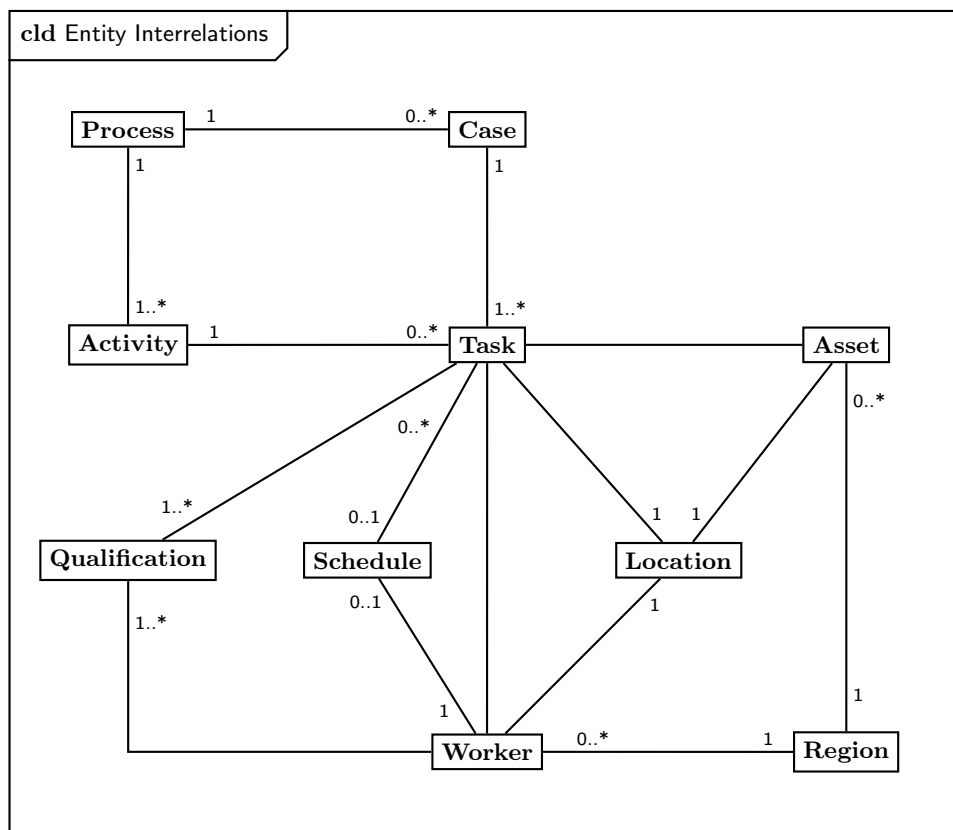


Figure 2.9: Class diagram of entity interrelations

$\sigma$  and any task  $\tau$  that belongs to  $\sigma$ , the function  $\text{pred}_\sigma(\tau)$  returns the predecessor of the task  $\tau$  in the schedule  $\sigma$ . For any schedule  $\sigma$  and any task  $\tau$  that belongs to  $\sigma$ , the function  $\text{succ}_\sigma(\tau)$  returns the successor of the task  $\tau$  in the schedule  $\sigma$ .

The UML class diagram in Figure 2.9 illustrates the interrelations of the entities of mobile work.

## 2.4 State Models of Entities

Based on the characterization of abstract mobile processes as illustrated in Figure 2.6 as well as the assignment and performance of mobile tasks as illustrated in Figures 2.7 and 2.8, state models for workers, tasks, and processes can be given. A preliminary version of the state models has already been published by the author in [51].

### 2.4.1 Mobile Workers

Figure 2.10 shows a UML state diagram of the states a worker can adopt. The state diagram corresponds to a typical working day of a mobile worker.

When the working day starts, the schedule of a worker is already created and assigned. At this point the worker reaches the state **Ready**. Considering a typical daily routine the worker enters the state **Traveling** as soon as he drives to the site of the first task of his schedule. After reaching the working site two alternative situations can occur: If the worker's task is preceded by another task of the respective case *and* this other task has not been finished yet, the worker has to wait until that task is finished, and he enters the state **Waiting**. Note that if the preceding task blocks the current task, the preceding task *must* have been assigned to a different worker and *may* be situated at a different site. For illustration purposes consider the process depicted in Figure 2.3. The blocked task could for instance be "Work @L2" while the blocking task could be "Off @L1".

If no preceding task is blocking a worker's current task, the worker starts to work on the task and enters the state **Working**. If the schedule of the worker is empty after finishing the task, the worker returns to the state **Ready** and then finishes his working day via the state **Not Ready**. If the schedule is not empty after finishing the task, the worker starts to travel to the next task, entering the state **Traveling** again. At any time of the working day the worker can be withdrawn from his current task. This may be due to the occurrence of cases with high priority, e.g., emergency situations. In such a situation a new task is added to the top of the worker's schedule, and the

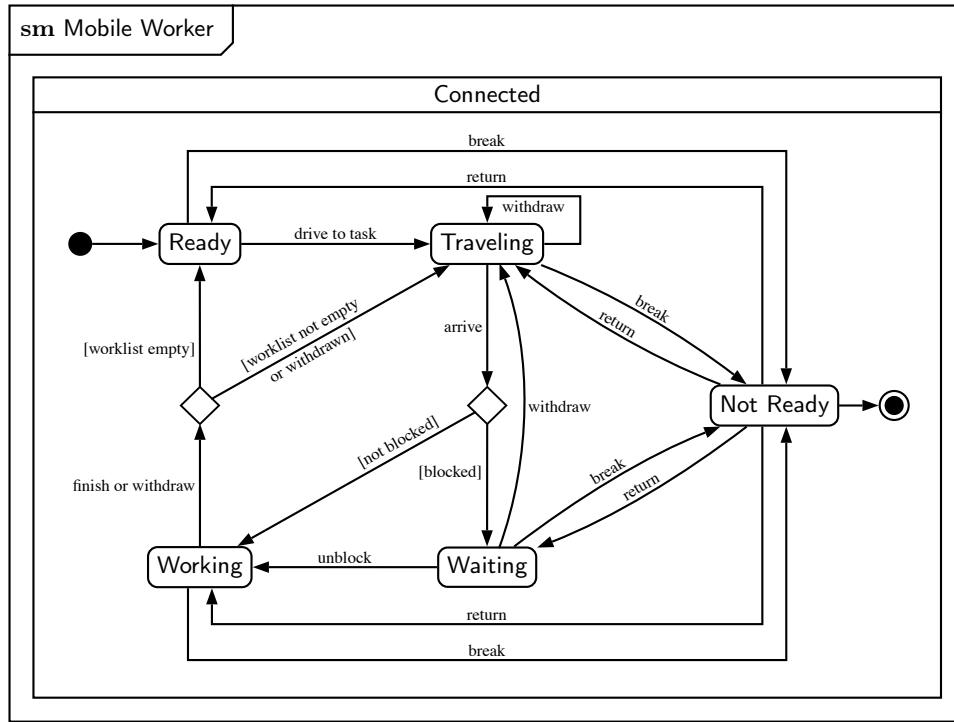


Figure 2.10: State of a mobile worker

worker interrupts his current activity to travel to the location of the newly added task. This is depicted in Figure 2.10 by the **withdraw** edges from the states **Traveling**, **Waiting**, and **Working**, by the **drive to task** edge from the state **Ready**, and by the **return** edge from the state **Not Ready**. From every state the worker can enter the state **Not Ready** via the **break** edges. This state may indicate a lunch break or a traffic accident, for instance. Note that due to the nature of mobile work a state change from the state **Not Ready** is only possible back to the preceding state of the worker.

All states depicted in Figure 2.10 that were discussed above are substates of the superstate **Connected**. The state **Connected** is introduced because the states are declared with no respect to the knowledge the headquarter of the organization has about a worker's state. Since mobile workers operate in the field with unreliable and unpredictable data or phone connections, the current state of a worker may be unknown to the headquarter at any time during operation.

In addition to the already introduced states, Figure 2.11 adds the state **Disconnected** to the set of a worker's possible states. The model introduced in this work assumes that every state change of a worker is reported to the headquarter via data communication. As soon as a timeout occurs, the headquarter's state machine describing the respective worker enters the

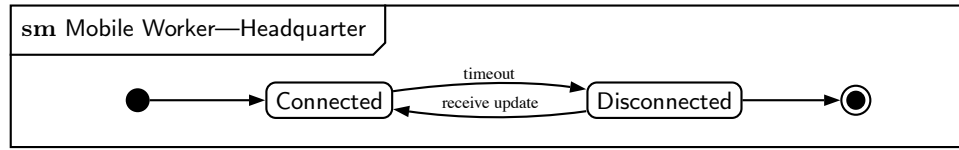


Figure 2.11: Additional information of worker's state at the headquarter

state **Disconnected**. The timeout period may be defined individually for every task or sequence of tasks due to different travel distances and working durations. During a period in which a worker is disconnected and cannot be reached; this implicates that he cannot be rescheduled by the workforce management system. Whenever an update from a disconnected worker is received by the headquarter, the information of the worker's state is updated accordingly.

### 2.4.2 Mobile Tasks

In conjunction with the state model of mobile workers, a state model of tasks can be defined. Figure 2.12 shows the UML state diagram of a mobile task.

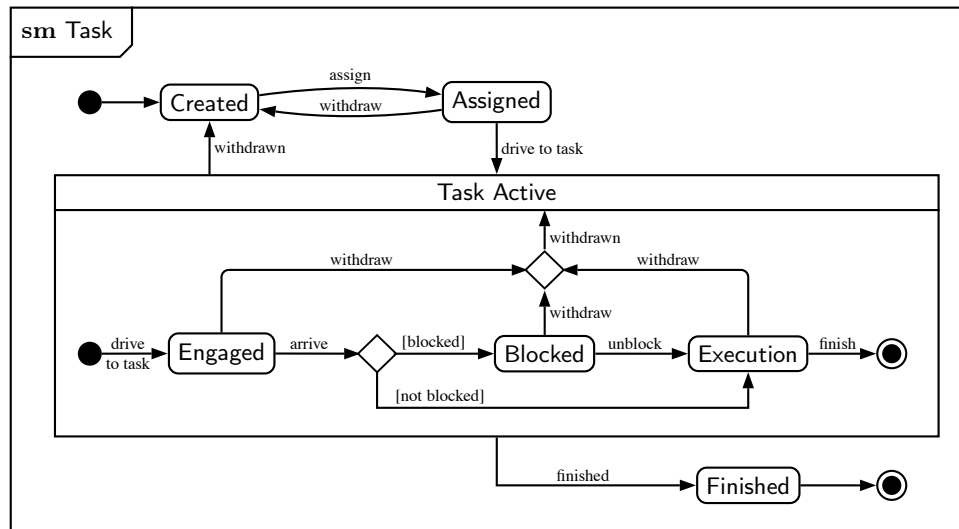


Figure 2.12: State of a task

Before a task is created at a working day, its originating case has already been chosen for execution. With the creation of a case all of its tasks are created alongside. The creation of a task is the entry point of that task to the state model. After its creation a task enters the state **Created**. As soon as the task is assigned to a worker and added to this worker's schedule, it enters the state **Assigned**. A task of the state **Assigned** can either be



withdrawn from the schedule by the scheduler and thus return to the state **Created** or the worker starts to travel to the location of this task, then the task enters the state **Task Active**. Note that a task remains in the state **Assigned** during the execution of preceding tasks of the same worker.

The state **Task Active** aggregates the states **Engaged**, **Blocked**, and **Execution**. As soon as the worker drives to the site of a task, the task's state implicitly changes to **Engaged**, meaning that this task is the current or topmost task in a worker's schedule. After the worker arrives at the site, the execution of the task can be either blocked by a preceding task or not blocked. The resulting states comply with according states of the worker after arrival. If the task is blocked, it enters the state of the same name, **Blocked**; otherwise it enters the state **Execution**. This complies with the worker entering either **Waiting** or **Working**, triggered by the same events. As soon as a formerly blocked task becomes unblocked, it enters the state **Execution** and the worker can start working. After finishing the work the task changes to the state **Finished**.

During all of the states aggregated by **Task Active**, a task can be withdrawn from the worker and in turn re-enter the state **Created**. This implies that the worker immediately stops to travel, wait, or work on this task and starts the next task of his schedule. The next task in the schedule might have been altered by the workforce management system during the withdrawal of the current task. Such situations usually occur if highly prioritized cases (e.g., emergencies) arrive in the system demanding immediate execution. It is impossible to withdraw a task after it was finished. Instead the whole case must be considered at a higher level of process control. This situation will be discussed later in this section.

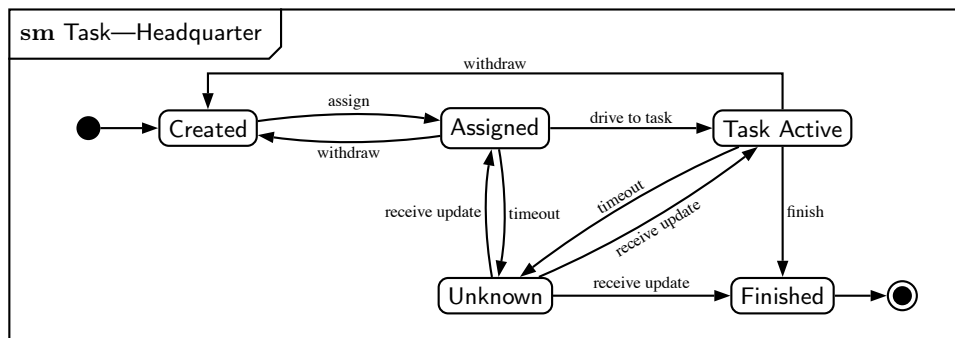


Figure 2.13: Additional state information about a task as seen by the headquarter

As for the state of a worker also the state of a task may be unknown to the headquarter controlling the mobile operations due to the loss of data or phone connections. This fact is considered in Figure 2.13. The relations between

the states **Created**, **Assigned**, **Task Active**, and **Finished** are identical to those depicted in Figure 2.12. The state **Task Active** is an aggregation of the states depicted in Figure 2.12. In addition to the task's state model discussed above, the state **Unknown** is introduced indicating that the headquarter cannot determine the current state of a task due to a connection timeout. Whenever a task is either in the state **Assigned** or in the state **Task Active** and the data connection to the worker is lost, the headquarter assumes the respective task to be in the state **Unknown**. Whenever an update from that worker is received, the headquarter's state model can be updated to the corresponding state. It is assumed that during a task being in the state **Unknown** this task might not be withdrawn from the worker to avoid unpredictable states of the whole case.

### 2.4.3 Mobile Cases

The state models of a task lead to the state model for a case, depicted in Figure 2.14.

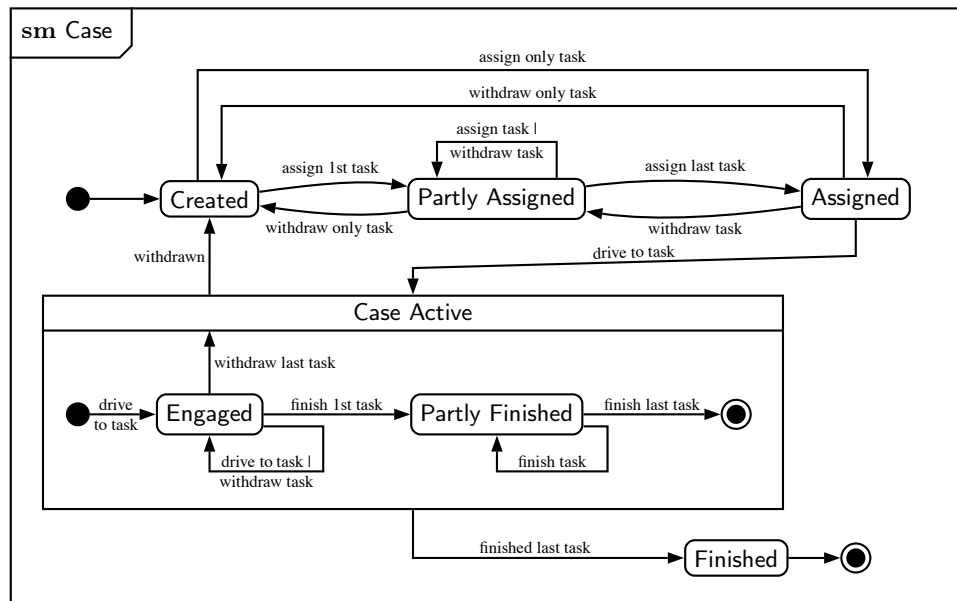


Figure 2.14: State of a case

As already stated above, the tasks of a case are created alongside with the creation of the case itself. Since a case is an ordered list of tasks, the state models of tasks and cases are quite similar. Thus, the change of the state of a task usually triggers a change of the state of the corresponding case.

It is obvious that with its creation a case enters the state **Created**. With the assignment of the first task to a worker, the case changes to the state **Partly**

**Assigned**, and with the assignment of the last unassigned task, it changes to the state **Assigned**. If the case contains just one task, the state of the case changes directly from **Created** to **Assigned**. Whenever a task is withdrawn from a worker as long as no worker started to work on one of the tasks of the case, the states change in the opposite direction accordingly. If the case is in the state **Partly Assigned** and tasks are withdrawn or assigned such that none of the aforementioned changes of states occur, the case remains in the state **Partly Assigned**—see the looped edge of the state in Figure 2.14.

After assigning the tasks, the workers may start to drive to the locations of their respective current tasks. Since the processes considered here demand the completion of all activities, the assignment of a case's tasks must be complete too—i.e., all tasks must be assigned to workers<sup>1</sup>. Thus, the assignment of work takes place and is completed before the actual planning period starts; and thus, no worker can start to drive before all schedules are complete. As soon as an arbitrary worker that has to execute at least one task of the case starts to drive to the task's location (the task's state changes to **Engaged**), the case's state changes to **Engaged** accordingly. The case remains in the state **Engaged** until either the first task of the case is finished or all tasks of this case have been withdrawn. After the first task is finished, the case enters the state **Partly Finished**. It is assumed that a case may or may not be withdrawn after one of its tasks is finished. This depends on the policy of the respective organization. To distinguish between both variants, finishing the first task of the case causes a change in the case's state. Otherwise a task could always be withdrawn after the first task of its case has already been finished. Note that Figure 2.14 depicts an organization where cases must be finished after their first task is finished. In contrast, organizations may allow late task withdrawal after finishing the case's first task. The state **Partly Finished** could then be omitted. If the last engaged task has been withdrawn, the state **Case Active** is left, and the case re-enters the state **Created** demanding a completely new assignment turn. As soon as the last task has been finished, the case enters the state **Finished**.

For the cases the same rules about the headquarter's knowledge of the case's state apply as for tasks and workers. Thus, in Figure 2.15 an extended state diagram is given for cases with regard to the uncertainty of data connections. All states and their interrelations are the same as in Figure 2.14 except for the state **Unknown**, which is added newly. The meaning of the state **Unknown** is similar to the according state for tasks (see Figure 2.13). Whenever a case is in one of the states **Assigned** or **Case Active** and the connection to an arbitrary worker involved in this case times out, the case enters the state **Unknown**. Additional timeouts and / or received updates keep the case

---

<sup>1</sup>In general it is also possible to schedule only a few tasks of a certain case. The execution of the remaining tasks of this case in the next planning periods could then be enforced by assigning the case with a high priority or with high individual downtime costs.

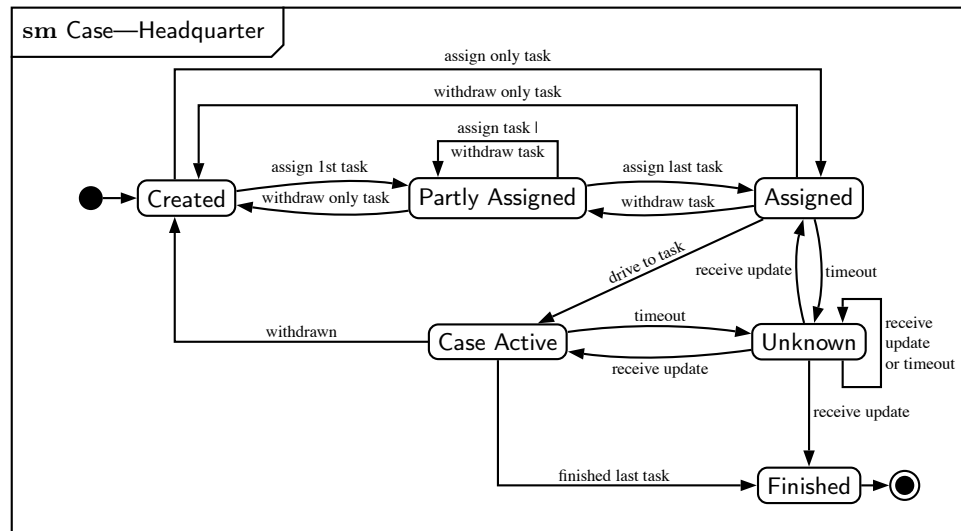


Figure 2.15: Additional state information about a case as seen by the head-quarter

remaining in the state **Unknown** until the states of *all* workers are known again—i.e., updates were received. The state of the case is then determined with regard to the according update messages.

## 2.5 Optimization Objectives

The general objective of all optimization efforts is to reduce the process environment’s overall costs.

As already stated in Chapter 1, this work is intended to present a business domain independent optimization method for mobile processes. It is thus not focused on the business related process properties. Optimization potentials bound to such properties are usually considered by domain experts, and such considerations are often already finished when further organizational and administrative optimizations are gone about. Thus, in this work, the business domain properties are not subject to change, and are not considered as being part of the optimization objectives. Thus, all properties of business processes bound to the business objectives are assumed to be fixed within the organization considered. Among such fixed properties of the business processes are:

- the hierarchical structure of the business processes,
- the execution order of activities within one and the same business process,

- the default execution times of activities and processes,
- the execution frequency of the processes, and
- the material and energy consumed by the execution of the processes.

The residual properties of mobility that are subject to change by optimization techniques are related to workers, travel, and process administration. Altering these properties for optimization reasons is incorporated into the answers to the following questions:

1. Which qualifications of the workforce are needed by the organization? The needs are defined by the frequency and the types of work to be accomplished.
2. Which distribution of qualifications among workers and regions matches the organization's minimal needs?
3. Which number of workers allows the enterprise to perform all business cases with an acceptable quality of service?
4. What impact has the modification of the dimensions of regions on the overall costs?
5. Which workforce scheduling method helps reducing process execution costs best?
6. Is the quality of the scheduling method influenced by the concrete organization or is it rather bound to mobility in general?
7. Which are the implications of insufficient process execution?
8. How can such implications be quantified?

Based on the aforementioned assumptions and questions, several types of costs are present in mobile process environments and will be discussed in the following sections.

### 2.5.1 Worker Related Costs

For workers of an organization two different types of costs can be specified. The first type is the worker's cost per time unit which is constantly incurred, regardless of the actions a worker performs at a certain moment. The second type of worker related costs are incurred by the fact that organizations need workers with certain qualifications to perform their business processes.

**Definition 2.16 (Basic costs)** *The basic costs of a worker are the fixed costs per time unit incurred by this worker. The basic costs of a worker arise from the worker's salary including all dues and taxes the organization has to pay for. The basic costs per time unit of worker  $w \in \mathcal{W}$  are a nonnegative value denoted by  $\kappa_w^{ub} \mid \kappa_w^{ub} \in \mathbb{N}^+$ .*

**Definition 2.17 (Qualification costs)** *The qualification costs of a worker are the costs per year incurred by the effort to preserve and to enhance this worker's qualifications. The qualification costs of worker  $w \in \mathcal{W}$  are a nonnegative value denoted by  $\kappa_w^q \mid \kappa_w^q \in \mathbb{N}^+$ .*

If we further assume the number of working time units per year of worker  $w \in \mathcal{W}$  to be denoted as  $n_w^u$ , we can define the worker's overall costs per time unit as

$$\kappa_w^u = \kappa_w^{ub} + \frac{\kappa_w^q}{n_w^u} \quad (2.1)$$

and the overall worker related costs per year as

$$\kappa^y = \sum_{w \in \mathcal{W}} (\kappa_w^u n_w^u). \quad (2.2)$$

### 2.5.2 Travel Related Costs

As traveling is performed recurrently in mobile process environments, the associated costs have a significant impact on the overall costs of mobile environments. Travel costs arise from both travel time and travel distance.

**Definition 2.18 (Task's travel time and distance)** *The travel time  $t_\tau^{travel}$  of a task  $\tau \in \mathcal{T}$  is the duration of a worker's journey from his current location  $l_w$  to the location of  $\tau$ . The current location of the worker is either its depot  $l_w^{depot}$  if  $\tau$  is the first task of this worker for the planning period or it is the location of the task preceding  $\tau$  in the worker's schedule  $s_w$ , otherwise:*

$$l_w = \begin{cases} l_w^{depot}, & \text{if } \nexists \text{ pred}_{s_w}(\tau) \\ l(\text{pred}_{s_w}(\tau)), & \text{otherwise} \end{cases}$$

The accordant travel distance to the location of task  $\tau$  is denoted by  $d_\tau$ .

**Definition 2.19 (Travel time costs)** *The travel time costs  $\kappa_\tau^{travel}$  of a task  $\tau \in \mathcal{T}$  to be performed by a worker  $w \in \mathcal{W}$  depend on the time this*

worker needs to travel to this task and the overall costs per time unit of the performing worker. The travel time costs of  $\tau$  are a nonnegative value denoted by  $\kappa_\tau^{ttravel} = t_\tau^{travel} \kappa_w^u \mid \kappa_\tau^u \in \mathbb{N}^+, elem_{\sigma_w}(\tau)$  is true.

**Definition 2.20 (Travel distance costs)** *The travel distance costs of a task depend on the distance that the performing worker has to cover traveling to this task and on a fixed factor per distance unit. The travel distance costs of a task  $\tau \in \mathcal{T}$  are a nonnegative value denoted by  $\kappa_\tau^{dtravel} = d_\tau \kappa^{ud} \mid d_\tau, \kappa^{ud} \in \mathbb{N}^+$  with  $\kappa^{ud}$  denoting the costs per distance unit.*

The resulting total travel costs per task are then calculated by

$$\kappa_\tau^{travel} = \kappa_\tau^{ttravel} + \kappa_\tau^{dtravel}. \quad (2.3)$$

According to the definitions given above a number of additional examination criteria can be defined as follows.

The total travel time of worker  $w \in \mathcal{W}$  per planning period is denoted as  $t_w^{travel}$  and is calculated by

$$t_w^{travel} = \sum_{elem_{\sigma_w}(\tau)} t_\tau \quad (2.4)$$

and the total travel time for one planning period is calculated by

$$t^{travel} = \sum_{w \in \mathcal{W}} t_w^{travel}. \quad (2.5)$$

### 2.5.3 Case Related Costs

Case related costs are costs that arise from the execution of business cases. Such costs can be calculated per case, depending on the workers involved, on the duration of work, on travel as well as waiting times, and on the violation of the case's time window.

**Definition 2.21 (Task's waiting costs)** *The waiting costs of a task are time related and denote the costs incurred by the necessity of a worker to wait at the task's location until all preceding tasks of the same case are finished. The waiting costs  $\kappa_\tau^{wait}$  of a task  $\tau \in \mathcal{T}$  performed by worker  $w \in \mathcal{W}$  are a nonnegative value denoted by  $\kappa_\tau^{wait} = (t_\tau^{start} - t_\tau^{arrive}) \kappa_w^u \mid \kappa_\tau^{wait} \in \mathbb{Q}^+, elem_{\sigma_w}(\tau)$  is true.*

**Definition 2.22 (Task's working costs)** *The working costs of a task are time related and denote the costs incurred by the time a worker needs to perform the task. The work costs  $\kappa_\tau^{work}$  of a task  $\tau \in \mathcal{T}$  performed by worker  $w \in \mathcal{W}$  are a nonnegative value denoted by  $\kappa_\tau^{work} = (t_\tau^{finish} - t_\tau^{start})\kappa_w^u \mid \kappa_\tau^{work} \in \mathbb{Q}^+, elem_{\sigma_w}(\tau)$  is true.*

The resulting total time and travel costs  $\kappa_\tau$  of a certain task can be calculated by

$$\kappa_\tau = \kappa_\tau^{travel} + \kappa_\tau^{wait} + \kappa_\tau^{work} \quad (2.6)$$

and, accordingly, the total time and travel costs  $\kappa_c^{tt}$  of a case  $c \in \mathcal{C}$  are calculated by

$$\kappa_c^{tt} = \sum_{\tau \in \mathcal{T}_c} \kappa_\tau. \quad (2.7)$$

Analogously, the particular traveling, waiting, and working times and costs per case  $c \in \mathcal{C}$  can be calculated by

$$t_c^{travel} = \sum_{\tau \in \mathcal{T}_c} t_\tau^{travel}, \quad t_c^{wait} = \sum_{\tau \in \mathcal{T}_c} t_\tau^{wait}, \quad t_c^{work} = \sum_{\tau \in \mathcal{T}_c} t_\tau^{work} \quad (2.8)$$

$$\kappa_c^{travel} = \sum_{\tau \in \mathcal{T}_c} \kappa_\tau^{travel}, \quad \kappa_c^{wait} = \sum_{\tau \in \mathcal{T}_c} \kappa_\tau^{wait}, \quad \kappa_c^{work} = \sum_{\tau \in \mathcal{T}_c} \kappa_\tau^{work}. \quad (2.9)$$

Additional costs may arise by the duration of a case. For illustration purposes consider the power line extension process introduced in Section 2.1.2. As long as the line is switched off no energy is sold to the customers incurring duration costs.

**Definition 2.23 (Case duration costs)** *Case duration costs are costs incurred from the start of the first task of a case until the finish of the last task of that case. The duration costs  $\kappa_c^{dur}$  of a case  $c \in \mathcal{C}$  are a nonnegative value denoted by  $\kappa_c^{dur} = \kappa_c^{udur}(t_c^{finish} - t_{\tau_0}^{start}) \mid \kappa_c^{udur} \in \mathbb{Q}^+$ , being the duration cost per time unit of  $c$ .*

Similar costs may arise from the existence of a case. For illustration purposes consider the damage search and repair process introduced in Section 2.1.3. The situation is similar to the one introduced before, but the existence costs arise as soon as the line is damaged.



**Definition 2.24 (Case existence costs)** *Case existence costs are costs incurred from the timestamp 0 until the finish of the last task of a case. The existence costs  $\kappa_c^{ex}$  of a case  $c \in C$  are a nonnegative value denoted by  $\kappa_c^{ex} = \kappa_c^{dur} t_c^{finish} \mid \kappa_c^{dur} \in \mathbb{Q}^+$ , being the duration cost per time unit of  $c$ .*

In addition to the costs arising from the case's execution times, costs may be incurred if the time window of a case is violated. Such costs can be interpreted as penalties or fines an organization has to pay for late delivery of services in the field. Concerning the example of the power utility such a situation is, e.g., the completion of a power line later than agreed upon in the respective contract, if the late delivery of power to an industrial customer is under contractual penalty.

**Definition 2.25 (Time window violation costs)** *If a case is finished after the end of its time window, costs may be incurred by contractual penalties. Such costs—denoted by  $\kappa_c^v$ —depend on the duration of the violation of the time window and on a fixed factor  $\kappa_c^{uv}$  bound to the case:  $\kappa_c^v = (t_c^{finish} - t_c^{max})\kappa_c^{uv}$ . Usually a fixed cost factor  $\kappa_p^{uv}$  may be given per process and passed to the according cases ( $\kappa_c^{uv} := \kappa_p^{uv}, c \in C$  being an instance of  $p \in \mathcal{P}$ ), but the factor  $\kappa_c^{uv}$  for a case can also be defined independently from the process.*

It is easy to see that time window violation costs can be utilized to model case existence costs by setting tight time windows. The resulting total costs  $\kappa_c$  of a certain case can be calculated by

$$\kappa_c = \kappa_c^{tt} + \kappa_c^{dur} + \kappa_c^{ex} + \kappa_c^v \quad (2.10)$$

#### 2.5.4 Optimization Parameters of Mobile Environments

Based on the entities of mobile work introduced in Section 2.3 and the optimization objectives introduced in Sections 2.5.1, 2.5.2, and 2.5.3, this section examines the relations between the entities and how these relations can be parameterized to build different scenarios for the operation of a mobile environment.

Since regions are key entities in mobile environments a brief overview of their structure and implications is given first. The regionalization influences the peculiarity of the criteria bound to the optimization objectives introduced above. Based on these insights measures for the alteration of the regionalization are developed.

It was already stated in Definition 2.13 that a region is defined by the assets and workers associated to it and by the location of its depot. It is thus feasible

to state that a region is a compound of assets, workers, and a depot. This coherency will be utilized to model the optimization parameters of mobile environments.

For the sake of manageability mobile organizations often split their area into several regions with independent responsibility for the business processes performed. Examples of such a setup of a German power supply are depicted in Figures 5.2 and 5.3 (p. 139) where the latter shows the detail marked in the former. As already introduced in Section 1.1.2, regions are subject to historical evolution, which lets examining them be a promising intention to improve the cost situation.

The following parameters influencing the performance of mobile environments were identified with respect to the domain model. They can be used to find potential changes to the mobile environment, which improve the performance of process execution.

**Location of the Depot (LD)** Given are the regions and the assets assigned to each region. Unknown are the locations of the depots of the regions. The location of the depot of a region  $r \in \mathcal{R}$  can be constituted in different ways. Choose the location  $l_r^{depot}$

**LD1** as the center of gravity of the assets  $e \in \mathcal{E}_r$  of the region  $r \in \mathcal{R}$ .

**LD2** such that the sum of the distances between the depot and the assets  $\sum_{e \in \mathcal{E}_r} dist(l_r^{depot}, e)$  is minimal.

**LD3** by visual selection (e.g. mouseclick).

**Closest Depot (CD)** Given are the locations of the depots and the assets. Unknown is the assignment of assets to regions, i.e. the borders of the regions. Assets  $e \in \mathcal{E}$  are assigned to regions such that each asset is assigned to the depot closest to it. This criterion can either be applied after relocating the depots with LD or independently. The assets are assigned to regions such that the distance  $dist(l_r^{depot}, e)$  is minimal for all assets  $e \in \mathcal{E}$  and regions  $r \in \mathcal{R}$ . The regions can be found as the Voronoi tessellation (see, e.g., [85]) of the depots.

**Equal Average Travel Effort (EATE)** Given are the locations of the depots and the assets. Unknown is the assignment of assets to regions, i.e. the borders of the regions. Assets  $e \in \mathcal{E}$  are assigned to regions such that the average travel effort in all regions is equal. This criterion can either be applied after relocating the depots with LD or independently. The assets are assigned to regions such that the weighted sum

$$\frac{\sum_{e \in \mathcal{E}_r} dist(l_r^{depot}, e)}{|\mathcal{W}_r|}$$

is equal for all regions  $r \in \mathcal{R}$ .

**Weighted Average Travel Effort (WATE)** Given are the locations of the depots, the assets, and the number  $n_e^{visits} \in \mathbb{Q}$  of visits per asset and time unit. Unknown is the assignment of assets to regions. Assets  $e \in \mathcal{E}$  are assigned to regions such that the average travel effort in all regions is equal. This criterion can either be applied after relocating the depots with LD or independently. The assets are assigned to regions such that the sum  $\sum_{e \in \mathcal{E}_r} \text{dist}(l_r^{depot}, e) n_e^{visits}$  is equal for all regions  $r \in \mathcal{R}$ .

**Equal Asset-Worker Ratio (EAWR)** Given are the regions with depots and assets assigned and the workers. Unknown is the assignment of workers to the regions. Workers  $w \in \mathcal{W}$  are assigned such that the quotient  $\frac{|\mathcal{E}_r|}{|\mathcal{W}_r|}$  is equal for all regions  $r \in \mathcal{R}$ .

**Equal Qualification Distribution (EQD)** Let  $n_a^r$  be the number of executions of the activity  $a \in \mathcal{A}$  in the region  $r \in \mathcal{R}$  in a given period of time (e.g. one year). Let further

$$q_r^{req} = (\sum_{a \in \mathcal{A}} (n_a^r q_{a_1}^{req}), \dots, \sum_{a \in \mathcal{A}} (n_a^r q_{a_m}^{req})), m = |\mathcal{Q}|$$

be a  $|\mathcal{Q}|$ -tuple representing how often each qualification is required in this region per year. Let further

$$q_r^{avail} = (\sum_{w \in \mathcal{W}_r} q_{w_1}^{avail}, \dots, \sum_{w \in \mathcal{W}_r} q_{w_m}^{avail}), m = |\mathcal{Q}|$$

be a  $|\mathcal{Q}|$ -tuple representing how often each qualification is available in region  $r \in \mathcal{R}$ . The workers  $w \in \mathcal{W}$  are assigned to the regions such that the element-wise quotients

$$\frac{q_r^{req}}{q_r^{avail}}$$

are equal for all regions  $r \in \mathcal{R}$ .

**Total Number of Regions (TNR)** Given are the assets. Based on the other criteria regions and depots are created from scratch. A Voronoi diagram with the assets being the Voronoi sites may be used to support the creation of the regions.

**Start Location (SL)** Given are the regions with assets, workers, and depots assigned. Given is further the home location of each worker. Choose the location where workers start their tour

**SL1** to be the depot of the region  $r \in \mathcal{R}$ .

**SL2** to be the home location of each worker. In this scenario a regular (e.g. weekly) meeting at the depot is usually necessary for social and business reasons and must be considered. The outcome can be compared to additional costs, e.g. additional cars and communication costs.

**Qualification Dependent Scheduling (QDS)** In environments with dynamic scheduling short response times might be of high importance. If specialists with unique qualifications exist they only might get work assigned close to the center of their region or close to equipment known for frequent emergencies.

**Process Execution Frequency (PEF)** Processes might be executed more often than the respective legal restrictions demand. Reducing the case count might decrease overall cost. In turn the repair costs of the respective assets may rise. The criterion variates the number of executions for certain process types.

**Additional Qualifications (AQ)** Workers get qualifications that accelerate their throughput or qualifications that are frequently required above availability. Outcomes can be compared to qualification costs.

**Scheduler Heuristic Selection (SHS)** Different scheduling heuristics are utilized for

**SHS1** certain regions.

**SHS2** the whole organization.

Note that an enterprise may define other or more parameters of their choice. In Chapter 5 a selection of these parameters is used for the validation of this work.

## 2.6 Chapter Summary

In this chapter a domain model of mobile work was developed. The entities, state models, and objectives introduced are common for a large number of mobile business processes regardless of their actual business domain. In the following chapters this domain model will be utilized to develop a business modeling method that hides the mobility-driven properties and restrictions of mobile processes from the business modeler. In doing so the business modeler can focus on the business-specific properties of the processes to model. Nonetheless, the modeling method is capable of introducing the mobility aspects into the process models developed, as will be demonstrated in Chapter 3. Besides being the basis for the following chapters, the model can also be seen as a general reference model of mobile work.

## Chapter 3

# Simulation of Mobility

In this chapter we introduce a simulation method for mobile business processes and the environments they are executed in. The method helps to judge different improvement scenarios in business process reengineering projects. It can thus be utilized to identify promising scenarios to be considered further in such projects. It consists of a CPN domain model representing the domain model introduced in Chapter 2, a CPN process model of mobile business processes, the Simple Mobile Process modeling Language (SMPL) derived from UML activity diagrams, and a transformation scheme to convert SMPL models to CPN process models. The CPN models are executable in a simulator and, thus, facilitate the method to provide simulation support in BPR projects. CPNs are a yet powerful but complex language the average business analyst is not familiar with. Therefore, the SMPL and an appropriate transformation scheme are provided to increase the usability of the method by allowing for the modeling of mobile business processes with less effort than CPNs. The first section is concerned with the proper conditions of simulation runs and the requirements regarding the simulation method. In the second section the CPN domain model is described in-depth, and the third section concentrates on the CPN process model to build mobile business processes for simulation. The fourth section introduces the SMPL as a subset of UML activity diagrams as well as the transformation scheme. The chapter concludes with a summary of the insights gained.

## 3.1 Introduction to Dynamic Analysis and Simulation

### 3.1.1 Problems Addressed

Simulation systems often provide formal models to describe the situation to examine. Such formalisms as, e.g., Petri nets provide an enormous expressive power to the modeler of a specific problem. Even more flexibility and expressiveness Petri nets gain from extensions as, e.g., FUNSOFT nets [30] or CPNs [55].

The drawback of Petri nets themselves and their empowering extensions lies in their expressive power since modeling real world processes requires a deep understanding of PNs' formalism and semantics and leads to quite large nets with tens and hundreds of model elements [120]. This prevents the average business user or business analyst from modeling complex business scenarios with PNs [113; 128].

To a certain degree mobile environments consisting of entities as presented in Chapter 2 can be evaluated statically. Metrics like the worker-to-asset-ratio per region or the average depot-asset-distance can help to find differences between regions. Such information helps to identify candidate regions for optimization efforts. Nonetheless, analyzing the mobile environment this way an enterprise is not capable of predicting the behaviour of the mobile environment under the dynamic conditions of the operation of such systems. We conceptualize dynamic conditions as operational situations demanding immediate reaction (e.g., combustion of a power substation) as well as situations preventing the operation of parts of the workforce (e.g., traffic accident with workers involved, sickness of workers). While the organization may know about the statistical distribution and frequency of such situations, there is usually no knowledge about their influence on short-term, mid-term, and long-term operation and cost effects. Additionally, organizations as, e.g., utilities may have to handle business processes with previously unknown progress and skill profiles.

Consider the damage search and repair process introduced in Section 2.1.3 (p. 15). Since the result of the search task depicted in Figure 2.5 determines the subsequent process steps, preliminary planning of the accordant cases is not suitable. The resulting sequence of planning and work execution cannot be analyzed in a static way but needs to be evaluated by dynamic simulation.

In addition to the situations introduced above, dynamic conditions arise if the scenario to be examined is no real scenario but the result of applying parametrization criteria as introduced in Section 2.5.4. For such scenarios, usually, no corresponding operational experience exists and thus the scenarios have to be simulated. The simulation method introduced here will thus

provide an analytical environment to evaluate the static modifications to mobile environments under conditions similar to real operation in mobile environments. The method allows for the simulation of a whole process environment with different cases of different processes being executed in one and the same simulation instead of considering the processes separately.

CPNs [55]—an extension of Petri nets [92]—were chosen for the development of the simulation method because they have a complete formal basis, and tools for the modeling and simulation are available free of charge.

### 3.1.2 General Execution of Simulations

Figure 3.1 gives an overview of the basic activities when using simulation for the analysis of a mobile scenario. It is assumed that the domain model and the processes have already been loaded into the simulator.

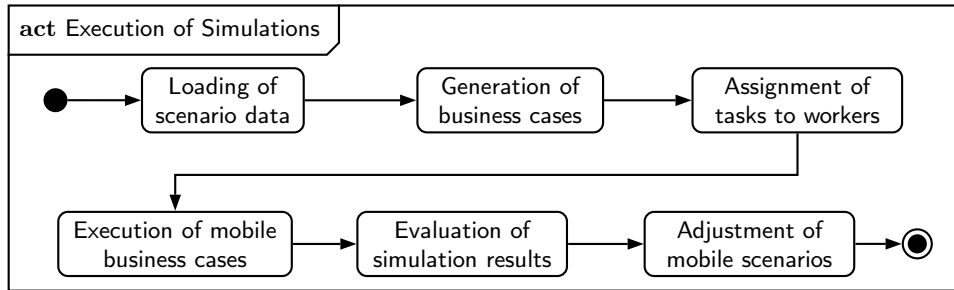


Figure 3.1: General execution of dynamic analysis with simulations

In the following, the activities are described in more detail:

1. Loading of scenario data imports all relevant data into the simulation model. Necessary data includes cases, workers, assets, and regions.
2. Generation of business cases: Based on how frequently the associated processes have to be executed, the loaded cases are assigned with priorities and arrival times. For the daily planning a list of cases is created while the urgent cases (see Definition 4.1, p. 88) arrive arbitrarily and as single cases.
3. Assignment of tasks to workers takes place in two different ways: During daily planning, a list of tasks is put into the daily schedules of the workers, representing their worklists. During the working day, randomly arriving urgent cases are added to the workers' schedules, postponing already planned tasks.
4. The execution of business cases represents the actual physical activity of the workers. During this step timestamps are adjusted, traveling

times, waiting times, and working times are accumulated and stored in the appropriate entities. As usual for simulations, the times are randomly distributed around the default values of the respective actions.

5. The evaluation of simulation results is splitted into the aggregation of necessary data by the simulation tool and the interpretation of the data by the user.
6. Based on the insights of the evaluation, the user might adjust the scenario's parameters and initiate a new simulation run.

## 3.2 Colored Petri Net Domain Model of Mobility

In this section the domain model of mobility introduced in Chapter 2 will be enhanced to an executable CPN domain model. The CPN domain model omits all *functional and process specific properties* and includes all *generic domain properties*. In this way, the domain model becomes reusable for different functional processes and different BPR projects. The freely available software CPN Tools<sup>1</sup> provides a stable basis for developing and executing CPNs and will thus be utilized for our purposes.

### 3.2.1 Entities of Mobile Work

Modeling the entities implements the definitions given in section 2.3. The very first version of the model's structure was inspired by a business process modeling approach introduced in [83]. The entities present in the CPN domain model are both representations of items of the real world (e.g., workers, tasks) and administrative items (e.g., list of workers, list of tasks). In CPNs entities are modeled as *color sets*, which are the CPN counterpart of classes in Object Oriented Programming (OOP). Accordingly, the terms *color* and *token* refer to objects in OOP. The modeling language is called CPN ML [18] and is an extension of the functional language Standard ML. CPN ML provides the typical simple types like `INT`, `STRING`, and `BOOL`. More complex color sets can be defined as records, in a similar way as `structs` are defined in the language C. The following color sets have been implemented to represent the entities of the simulation model. The names of the color sets match the names of the entities introduced in Section 2.3.

IDs are used to distinguish different colors of the same type. They are defined as strings:

```
colset ID = STRING;
```

---

<sup>1</sup>see <http://cpntools.org>



The color sets representing entities with static geographic context are locations, regions, and assets.

```
colset Location = record lon: INT          (* longitude *)
                    * lat: INT;            (* latitude *)

colset Region   = record id: ID
                    * ob: Location;        (* depot *)

colset Asset    = record id: ID
                    * loc: Location
                    * rid: ID;             (* region ID *)
```

The color set for an activity defines only its default execution time and the required qualifications.

```
colset Activity = record id: ID
                    * dt: INT              (* default execution time *)
                    * rq: LQualification;  (* required skills *)

colset Qualification = int with 0..1;

colset LQualification = list Qualification;
```

A task consists of the ID of the case, the activity, its precedence, the time window, the asset, and the timing information which is defined in a separate color set. The timing information (*TTiming*) is stored to evaluate the simulation results.

```
colset Task      = record id: ID
                    * cid: ID              (* case ID *)
                    * act: Activity
                    * prec: INT            (* precedence *)
                    * tw: TimeWindow
                    * ass: Asset
                    * tm: TTiming;

colset LTask     = list Task;

colset TimeWindow = record s: INT * e: INT; (* start / end *)

colset TTiming   = record work: INT        (* duration of work *)
                    * wait: INT           (* duration of waiting *)
                    * start: INT          (* timestamp of start *)
                    * finish: INT;        (* timestamp of finish *)
```

Processes consist of their respective activities, the execution frequency, and the number of daily emergency situations requiring the execution of this process.

```

colset Process = record id: ID
    * pa: LProcessActivity
    * ef: INT          (* frequency in percent *)
    * emfq: INT;      (* daily freq. of emergencies *)

colset ProcessActivity = record aid: ID          (* activity ID *)
    * prec: INT;      (* precedence *)

colset LProcessActivity = list ProcessActivity;

```

A case holds the information of its process, priority, task, and time window. The timing information (*CTiming*) of a case is not stored in a case itself but gathered from its tasks after they are finished.

```

colset Case = record id: ID
    * pid: ID          (* process ID *)
    * prio: INT        (* priority *)
    * t: LTask         (* case's tasks *)
    * tw: TimeWindow;

colset CTiming = record id: ID
    * appear: INT      (* timestamp of appearance *)
    * start: INT       (* timestamp of start *)
    * finish: INT      (* timestamp of finish *)
    * work: INT        (* cumulated duration of work *)
    * wait: INT        (* cumulated duration of waiting *)
    * trvl: INT;       (* cumulated duration of travel *)

```

A schedule consists of a list of its tasks, its planned duration, and a flag indicating that it was updated during the working progress. A worker holds information about its qualifications, current location, schedule, its timing, and the region it is assigned to. The timing information (*WTiming*) holds cumulated and current values about working, waiting, traveling, and overtime.

```

colset Schedule = record t: LTask
    * pd: INT          (* planned duration *)
    * dynup: BOOL;     (* true if updated dynamically *)

colset Worker = record id: ID
    * aq: Qualifications (* acquired skills *)
    * loc: Location      (* current location *)
    * sched: Schedule
    * tm: WTiming
    * rid: ID;          (* region ID *)

colset WTiming = record stmp: INT          (* last time stamp *)
    * work: INT        (* cumulated working time *)
    * wait: INT        (* cumulated waiting time *)
    * trvl: INT        (* cumulated travel time *)

```

```

* lastwork: INT    (* working time of prev. task *)
* lastwait: INT    (* waiting time of prev. task *)
* lasttrvl: INT    (* travel time of prev. task *)
*      ovr: INT;      (* cumulated overtime *)

```

Further color sets were defined for supporting reasons and are introduced as appropriate.

### 3.2.2 Model Overview

The CPN domain model consists of several hierarchically associated parts, which represent different aspects of the mobile resource-constrained environment. The model consists of a main part (**MobileEnvironment**) containing subparts of which the most important are the **Scheduler** and the mobile workers' state model (**ResourceState** and **MobileWork**). The general hierarchical structure of the domain model is as follows:

```

▽ MobileEnvironment
  ▷ LoadData
  ▷ Generator
  ▷ Scheduler
  ▷ ResourceState
  ▷ ClearUpdatedWorker
  ▷ GatherResults
▽ Process 1
  ▷ MobileWork(1)
  :
  ▷ MobileWork(n)
  :
▽ Process m
  ▷ MobileWork(k)
  :
  ▷ MobileWork(1)

```

The **MobileEnvironment** part contains references to the subparts according to the listing above. In parallel to the **MobileEnvironment** an arbitrary number of processes may exist. These processes are constructed out of instances of the **MobileWork** part of the model. By this design the model is split in a reusable part and a non-reusable, project-dependent part. The project-dependent part of the domain model contains only the CPNs representing processes. Even the **MobileWork** parts belong to the reusable model aspects. In the following the reusable part of the model is described in-depth while the project-dependent part involving the construction of process models out

of the reusable parts is described in Section 3.3. Figure 3.2 shows the top level of the CPN model.

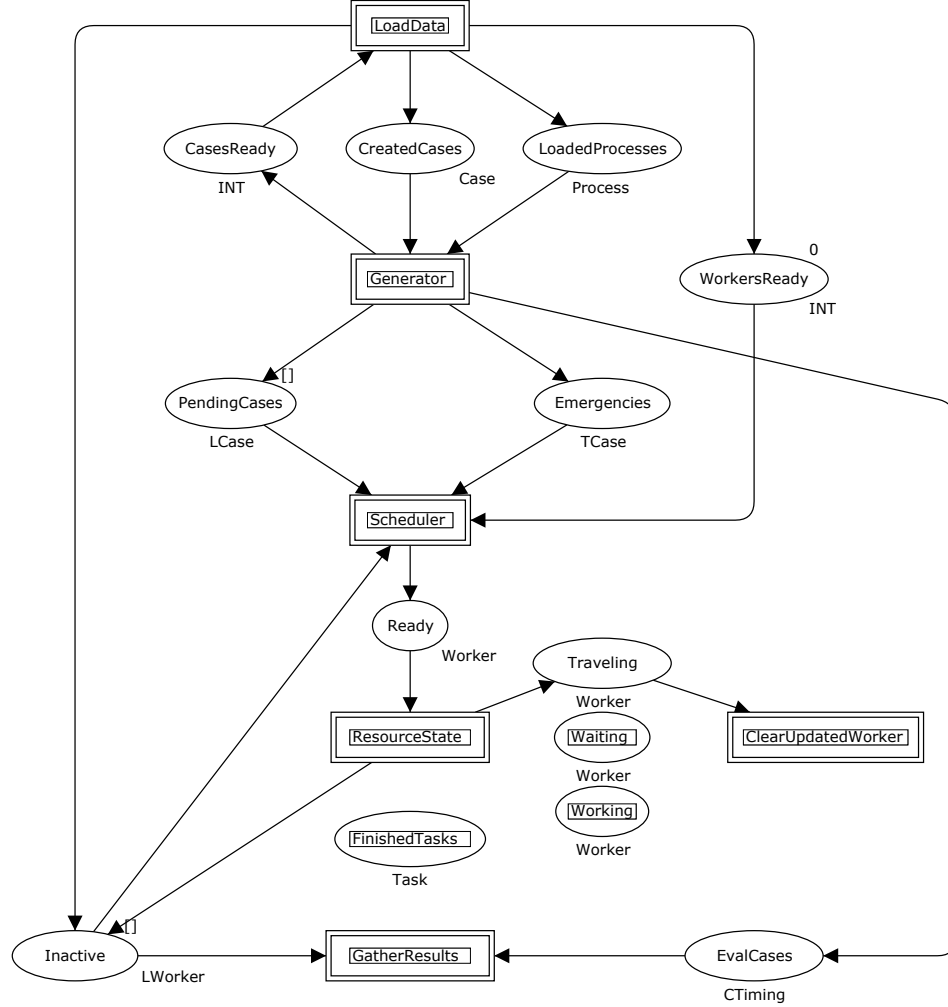


Figure 3.2: Model part **MobileEnvironment** (top level)

The transitions with double-lined borders mark submodel parts which are also called subpages. Places connecting subpages are ports. The annotation of a place shows the type of tokens it can hold (the color set) and, optionally, the initial marking. For each incoming edge a port has a corresponding place of the same type in the connected subpage. This corresponding place is marked as output port—**Out**—in the CPN of the subpage. Analogously, for each outgoing edge a port has a corresponding place of the same type in the connected subpage. This corresponding place is marked as input port—**In**—in the CPN of the subpage. Figures 3.2, 3.3, and 3.9 illustrate this: The place **CreatedCases** of the type **Case** represents the test cases loaded

into the simulator after having been enriched with additional data. The corresponding output place in the model part **LoadData** (Figure 3.3) and the corresponding input place in the model part **Generator** (Figure 3.9) are also named **CreatedCases**. The identical naming of ports and corresponding places is not required but increases the readability of the CPNs. This design of the model allows for the transfer of entities (e.g., cases) from one subpart to another. The second way to interconnect subparts are fusion places. While for ports at least two places must exist in different subparts and must be connected by port identifiers as well as by edges on the parent model part, fusion places represent exactly one place that can be referenced throughout the whole model regardless of the hierarchical depth of the subparts involved. Fusion places are marked by a frame identifying the name of the fusion (e.g., **FinshedTasks** in Figure 3.2).

**LoadData** initializes the simulator by loading the data of the mobile scenario from text files into certain places of the net. This data will then be used by the other parts of the model. The **Generator** generates work items, which will be assigned to mobile resources by the scheduler. The **Scheduler** assigns work items so that each resource has a daily working schedule. After having a complete schedule, a worker begins to flow through the **ResourceState** part of the model conducting the assigned activities. The number of work items and workers is only limited by the capacity of the system running the simulation. The model parts **ClearUpdatedWorker** and **GatherResults** implement supporting functions introduced later in this section.

### 3.2.3 Loading and Initializing Simulation Data

The submodel **LoadData** is depicted in Figure 3.3. It is responsible for initializing the simulation run. This includes to load the simulation data from text files, to initialize the test cases, and to initialize global data fields. The entities of the domain model contain numerous attributes with dynamic context (data that is not created before the simulation run), which need not to be loaded from the import files. Furthermore, entities have attributes that contain redundant data (e.g., depot locations are held in both **Workers** and **Regions**). The redundancy is necessary to perform the simulation runs—otherwise entities had to be aggregated several times during simulation—but makes the editing of simulation data a challenging task. Thus, the entities **Task**, **Case**, and **Worker** have corresponding input data types—**InTask**, **InCase**, and **InWorker**—with fewer attributes.

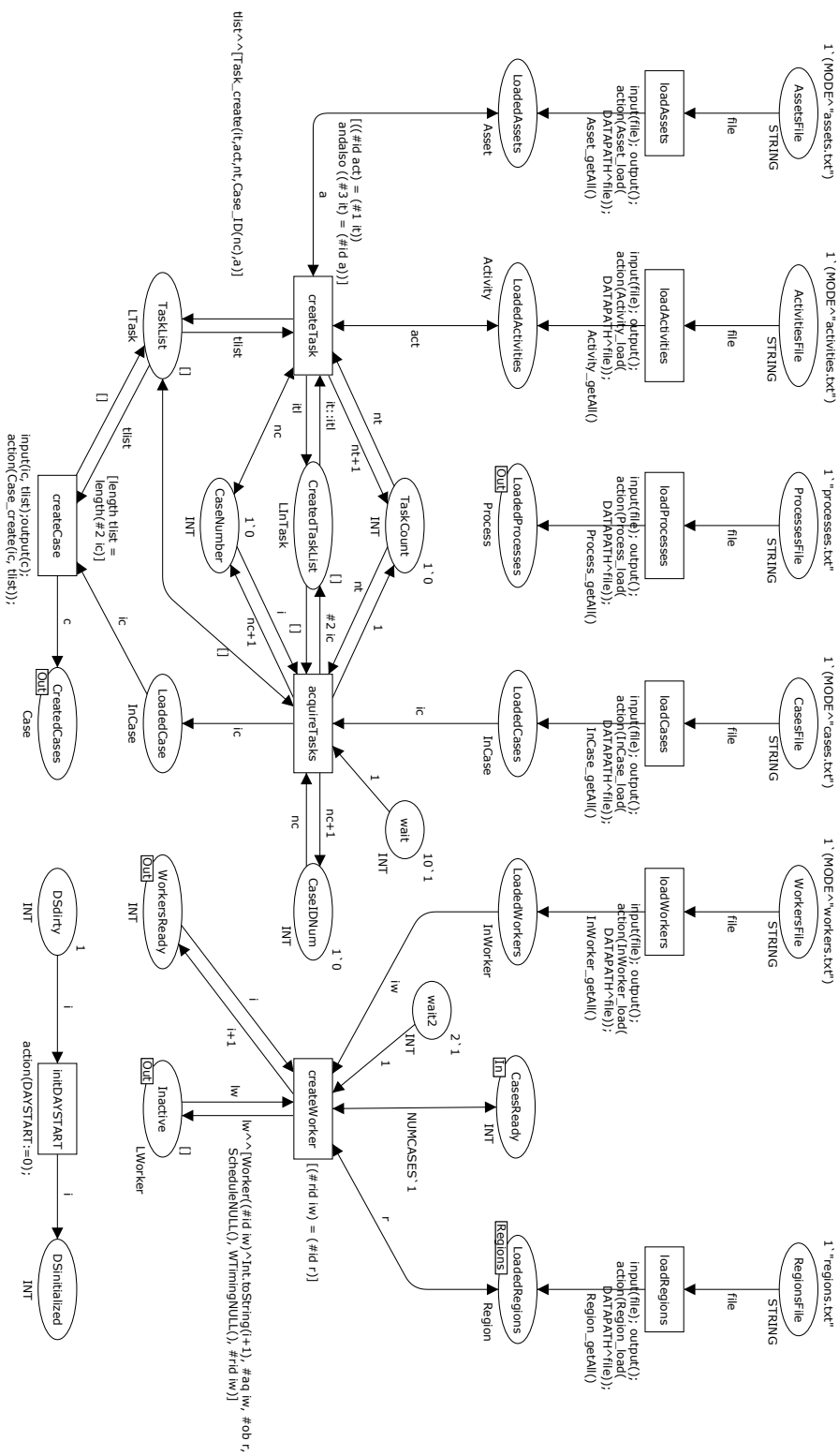


Figure 3.3: Model part LoadData

The input data color sets are defined as follows:

```
colset InTask = product ID          (* ID of the Task's Activity *)
                    * INT           (* precedence of the Task *)
                    * ID;           (* ID of the Task's Asset *)

colset LInTask = list InTask;

colset InCase = product ID          (* ID of the Case's Process *)
                    * LInTask;       (* List of Case's Tasks *)

colset InWorker = record id: ID
                    * aq: LQualification
                    * rid: ID;        (* ID of the region *)
```

Note, that CPN ML products are very similar to records, except that the components are not named. The components of a product are accessed by numbers, #1 referencing the first component.

To create the real entities the input data must be combined with input data from different entities and also enriched with preliminary data (e.g., for the timing attributes). Loading the input data is straightforward: Assets, activities, processes, cases, workers, and regions are loaded into the accompanying `Loaded...` places by entity-specific functions (e.g., `InWorker_load`, `Region_load`). The `Cases` and `Workers` need further handling to create the respective entities.

Once the case stubs are loaded the tasks are extracted from the `InCase` by the transition `acquireTasks` and transferred to the place `CreatedTaskList`. The subsequent transition `createTask` modifies each `InTask` of that list. It filters the correct `Activity` and `Asset` by matching it to the IDs stored in the `InTask` it and creates a new `Task`. This is achieved by the inscription function `Task_create`. The created task is added to the list in place `TaskList`. As soon as all tasks of the case are modified, the `Case` is created from both the list of tasks and the originally loaded case in the transition `createCase`. The cases are stored in the output port place `CreatedCases`, which makes them available in the submodel `Generator`. The places `CaseIDNum`, `CaseNumber`, and `TaskCount` are necessary for the creation of unique IDs of the respective entities.

After the `Generator` added all created cases to the list of pending cases (see Section 3.2.4) the loaded workers are initialized. Whether all cases are loaded is watched by the fusion place `CasesReady` and the inscription `NUMCASES'1` which controls the activation of the transition `createWorker` based on the number of expected cases for the simulation run. This number is stored in the variable `NUMCASES`. The workers are initialized by means of the function `Worker` of the transition `createWorker`. This incorporates assigning the appropriate region's depot (attribute `ob`) and creating an empty schedule as

well as empty timing information. Furthermore, a unique ID is created for each worker. The created **Workers** are added to the list of inactive workers in the fusion place **Inactive**. The number of created workers is stored in the fusion place **WorkersReady**. This information is used in the submodel **Scheduler** to enable the scheduler only after all workers are loaded and initialized.

The input files are human readable text files containing the entities as CPN ML colors. Figures 3.4, 3.5, 3.6, 3.7, and 3.8 show examples of input files for processes, activities, regions, assets, and workers. The operator ++ concatenates colors to multisets, which are markings of places. The simulation data can thus be loaded into the model directly as markings, without performing transformation steps. We developed a software tool for manipulating and exporting simulation input data. It is presented in Section 5.4.

```
1'{id="Station Maintenance 2 Lines",
  pa=[{aid="off1", prec=1},
      {aid="off2", prec=1},
      {aid="service_station", prec=2},
      {aid="on1", prec=3},
      {aid="on2", prec=3}],
  ef=10, emfq=0}++
1'{id="Customer Site Service",
  pa=[{aid="service_site", prec=1}],
  ef=3, emfq=2}
```

Figure 3.4: Example of an input file for processes

```
1'{id="off1", dt=900, rq=[1,0,0,0]}++
1'{id="service_station", dt=7200, rq=[1,1,0,0]}++
1'{id="service_site", dt=3600, rq=[0,1,0,0]}
```

Figure 3.5: Example of an input file for activities

```
1'{id="BZ0st", ob={lon=1446749, lat=5121473}}++
1'{id="BZSued", ob={lon=1448462, lat=5106963}}
```

Figure 3.6: Example of an input file for regions

```
1'{id="U5314", loc={lon=1431841, lat=5128354}, rid="BZ0st"}++
1'{id="U1205", loc={lon=1447563, lat=5109504}, rid="BZSued"}
```

Figure 3.7: Example of an input file for assets



```

1'{'id="M2", aq= [1,1,1,0], rid= "BZ0st"}++
1'{'id="BM1", aq= [0,1,1,1], rid= "BZSued"}

```

Figure 3.8: Example of an input file for workers

### 3.2.4 Generating Cases and Tasks

The purpose of the **Generator** is to create test cases that match the organization's real situation for the scenario provided. For creating the test cases the occurrence rate of the different processes has to be considered. As it was already defined above, the occurrence of a process is called a case. The occurrence rates include both the frequency of cases planned before the planning period and the frequency of cases emerging as urgent cases. In the realm of utilities it is, e.g., possible that the interchange of a power station's transformer is a planned case due to a changing energy demand at that station or as an emergency case that is initiated by a burned transformer.

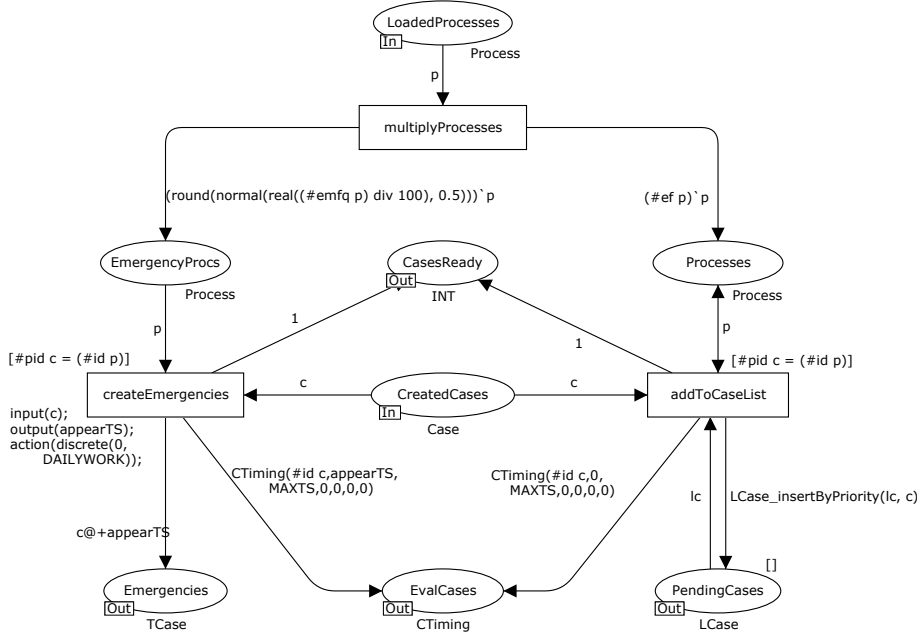


Figure 3.9: Model part Generator

The submodel **Generator** is depicted in Figure 3.9. As shown in Figure 3.4, for every process exactly one specimen is loaded ( $1'\{\dots\}$ ). The respective frequencies of occurrence are specified in the attributes **ef** and **emfq**. The attribute **ef** holds the relative frequency of a process compared to all other processes. Consider, for example, the following process stubs:  $p1=\{\dots, ef=80, \dots\}$ ,  $p2=\{\dots, ef=60, \dots\}$ ,  $p3=\{\dots, ef=60, \dots\}$ . The re-

sulting percentages of occurrence are 40%, 30%, and 30% for `p1`, `p2`, and `p3`, respectively. By specifying the frequencies of occurrence this way, it is possible to use any absolute number of occurrences in the set of processes (e.g., per week, per day, per year). The attribute `emfq` holds the average absolute number of daily occurrences of the process as an emergency situation multiplied by 100. The value is multiplied by 100 to avoid real numbers in the input, which cannot be handled by the import functions of CPN Tools. The transition `multiplyProcesses` creates copies of the loaded processes in quantities according to the occurrence attributes. The places `Processes` and `EmergencyProcs` hold the copies of the process tokens. The transition `addToList` creates a list of cases to be added to the daily schedules of the workers by the submodel `Scheduler`. The relation between the different types of cases is assured by the relation of process tokens in the place `Processes` and CPN Tools' random selection of tokens to be consumed by transitions. The number of tokens in the place `EmergencyProcs` is normally distributed with a mean of `emfq` and a variance of 0.5 with respect to the fact that `emfq` is the average number of daily occurrences. The transition `createEmergencies` creates the according cases with a randomly chosen time stamp and stores them in the place `Emergencies`. The places `PendingCases` and `Emergencies` are output ports from where the case tokens are transferred to the submodel `Scheduler` to be assigned to the workers' schedules.

### 3.2.5 Scheduling the Workforce

In mobile processes the assignment of workers and tasks is restricted by both the skill-match of workers and tasks and by their respective geographic location. The purpose of the `Scheduler` (see Figure 3.10) is to provide the assignment of `Task` tokens to the schedules of `Worker` tokens. Although the function of the scheduler is basically to assign work, it is split into two parts: the daily planning of tasks where schedules for the workers' whole work day are created (top half of Figure 3.10) and the reaction to urgent cases. The latter is depicted in the bottom half of Figure 3.10.

Basically, both functions work in the same way and differ only in a few aspects. The transaction `planCase` takes the set of all inactive workers `lw` from the fusion place `Inactive` and the first case `c` from the place `PendingCases` which holds a list of all cases to be executed, ordered by priority (highest priority first). After the `Worker` tokens have been loaded and initialized, they are held by the place `Inactive` (see Figure 3.3 `LoadData`). The function `Case_plan` inserts the tasks of `c` into the schedules of the appropriate workers. Cases not scheduled due to full workers' schedules are returned into the place `PendingCases`. For performance and maintenance reasons the scheduling algorithms are implemented as an external application in Java (see Section 5.2). The function `Case_plan` transfers the case to the Java

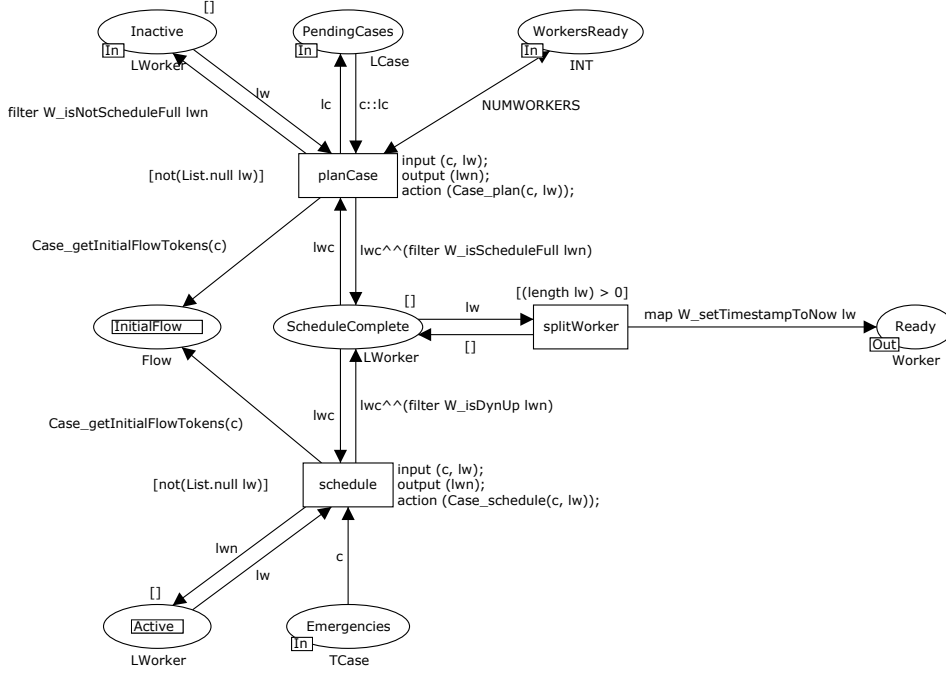


Figure 3.10: Model part Scheduler

program via TCP and handles the result accordingly. The scheduling software was developed by Carolin Ulbricht in her master's thesis [117] and the TCP interface was developed by Alex Klebeck in his master's thesis [65]. The scheduling algorithms are discussed in Section 4.6.4. The input port place **WorkersReady** and the inscription of the connecting edge assure, that the assignment of tasks starts only after all workers are initialized by the submodel **LoadData**.

Analogously, the transaction **schedule** takes the set of all active workers from the fusion place **Active** and an urgent case from the place **Emergencies**. Note that the place **Emergencies** holds tokens of type **TCase**, which implies timed **Case** tokens. Thus, the transaction **schedule** is activated whenever the simulator reaches the lowest timestamp of the tokens in the place **Emergencies**. Workers are assumed to be active—and, accordingly, associated tokens are held in the place **Active**—whenever they are in one of the states *Traveling*, *Waiting*, or *Working* of the state diagram in Figure 2.10 (p. 25). The cases are assigned by the function **Case\_schedule** of the transaction **schedule** in the same way the function **Case\_plan** works, i.e., they are transferred to the external Java application.

Every executed task consumes and produces a **Flow** token. These tokens are utilized to ensure the correct flow of workers through the process model and, thus, the execution of the case's tasks in the correct order. The integration of

process models into the simulation model is discussed in detail in Section 3.3. Whenever a case is planned or scheduled, the process flow must be initialized by creating the input **Flow** tokens for the case's tasks with precedence 1. This is achieved by the function **Case\_getInitialFlowTokens**, and the resulting tokens are stored in the fusion place **InitialFlow**.

Whenever a worker's schedule is complete, the modified list of workers **lwn** is filtered and the matching workers are transferred to the place **ScheduleComplete**. For planning cases a schedule is complete as soon as it holds tasks with an estimated working and traveling effort of **DAILYWORK** or above. **DAILYWORK** is a global variable representing the daily duty of a worker in seconds. It is set to 28,800 seconds (8 hours) by default but can be modified to fit the organization's needs. This criterion is filtered by the function **W\_isScheduleFull**. For the scheduling of urgent cases a schedule is complete as soon as the whole case is scheduled *and* the schedule was altered. These criteria are filtered by the function **W\_isDynUp**, which simply checks whether the flag **dynUp** of the according **Worker** token is set to **true**. Whenever the function **Case\_schedule** alters the schedule of a worker to insert a task of an urgent case, it sets this flag to **true**. Scheduling concludes with the transition **splitWorker**, which retrieves the workers from the list **lw** as single tokens for the execution of mobile work in the submodel **ResourceState**. The workers are kept in lists in the submodel **Scheduler** to ease the implementation of the domain model. The usage of lists enables the transfer of whole sets of workers, which is necessary since the scheduling algorithms need to access all workers.

### 3.2.6 State Model of Mobile Workers

The state cycle of a worker is represented by the submodels **ResourceState** (see Figure 3.11), **ClearUpdatedWorker** (see Figure 3.12), and **MobileWork** (see Figure 3.13). The submodel **ResourceState** handles the state related functionality common to all workers as traveling, finishing of the daily duty, and updating the schedules after tasks are finished. The submodel **MobileWork** in opposition handles all process specific functionality as waiting for the finishing of preliminary tasks of the same case as well as the actual work. The submodel **ClearUpdatedWorker** was introduced for increased readability. It just removes workers, whose schedules were updated with urgent cases, from the pool **Updated**.

Whenever a worker's schedule is completed by the **Scheduler**, the **Worker** token shows up in the place **Ready**. The transition **startTravel** transfers the tokens from the place **Ready** to the place **Traveling**. The functions **W\_travelToNextLocation** and **W\_timeToNextLocation** adjust the worker's location attribute **loc** and its timestamp, respectively. The pools **Active**

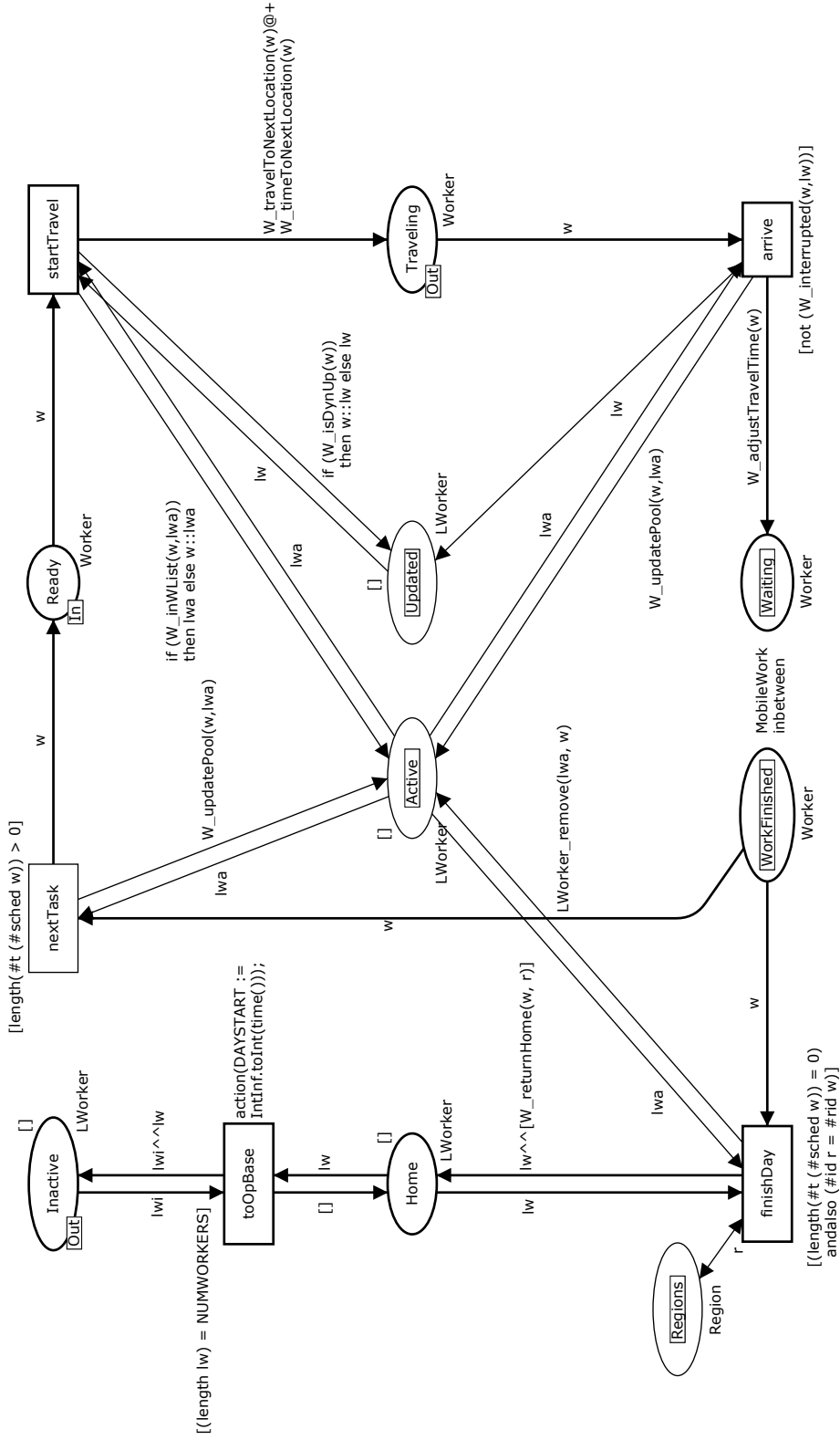


Figure 3.11: Model part ResourceState

and **Updated** represent the connection of a mobile worker to the central work control unit (submodel **Scheduler**) of its organization and enable the handling of this connection. The scheduler can contact workers by accessing them in the fusion place / pool **Active** whenever it has to adjust a schedule for whatever reason. The pool **Updated** holds all tokens of workers that have been rescheduled. This enables to remove tokens that were held in one of the places **Traveling**, **Waiting**, or **Working** during rescheduling. For the interaction of the transition **startTravel** with the pools **Active** and **Updated** the following rules apply:

1. The worker starts its working day with a newly created schedule and has thus to be added to the **Active** pool. This is ensured by the function **W\_inWList**, which checks if the list **lwa** already holds a token with the identical **id**, if not, the **Worker w** is added to the head of the list. The **Updated** pool remains unchanged.
2. The worker continues his work after having finished at least one preliminary task of his schedule. He thus already is in the **Active** pool and must not be added to it. The **Updated** pool remains unchanged.
3. The worker continues his working day after having been assigned with a task of an urgent case. In this case the workers' flag **dynUp** was set by the scheduler and the token has to be added to the **Updated** pool. This is ensured by the function **W\_isDynUp**. The **Active** pool remains unchanged.

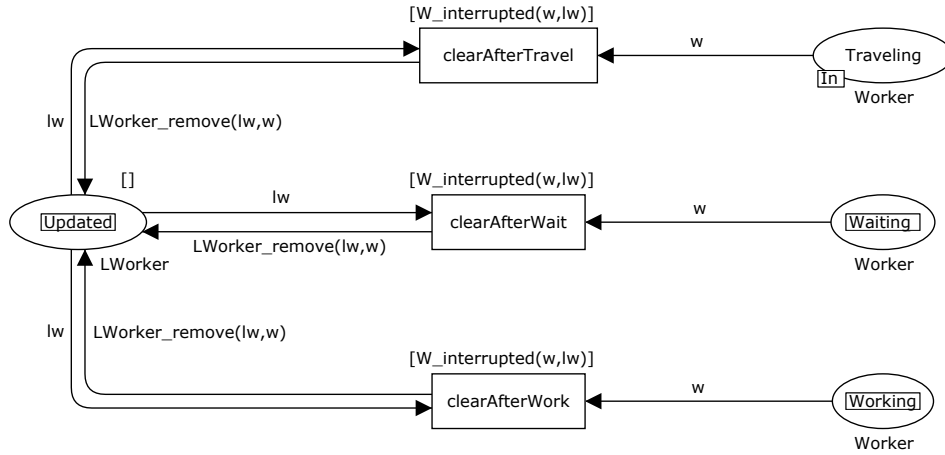


Figure 3.12: Model part ClearUpdatedWorker

The model time of CPN Tools is increased steadily but not by a constant value. Instead its progress depends on the timestamps of the tokens in the

net. Whenever a state is reached where no transition is activated (can fire), the model time is set to the timestamp of the token with the smallest timestamp larger than the current model time. As long as a token has a timestamp that is higher than the current model time of the simulator, this token cannot influence the behavior of the simulation. It just stays in its current place and is treated as if it was not existing. Whenever a timestamp of a timed token is reached by the simulator, this token becomes available for the activation of transitions.

As a token is transferred to the place **Traveling**, the accordant travel time is added to the timestamp of the token by the function `W_timeToNextLocation` based on the distance between the worker's current location in its attribute `loc` and the location of the first task in its schedule (attribute `sched`). As long as the simulator does not reach the timestamp of the token, it stays in the place **Traveling**.

The transition **arrive** is activated only if the incoming token `w` represents a worker who has not been rescheduled during his stay in the place **Traveling**. Otherwise the transition **clearAfterTravel** of the submodel **ClearUpdatedWorker** (see Figure 3.12) is activated. This is ensured by the function `W_interrupted`.

Whenever the scheduler assigns a new task to a worker from the pool **Active**, it creates a new **Worker** token for this worker and forwards it to the place **Ready**. Thus, two tokens representing the same worker are present in the state cycle of which the first (or outdated) one has to be removed from the simulation. This is done by the consumption of the token by the `clearAfter...` transitions of the submodel **ClearUpdatedWorker**. Further, the copy of the newly created token for this worker has to be removed from the pool **Updated** since the reason for its existence in this pool is to remove its outdated alter ego. This is achieved by the function `LWorker_remove`.

If the token `w` leaving the place **Traveling** does not represent an outdated worker the transition **arrive** is activated and the token is forwarded to the fusion place **Waiting**. The function `W_adjustTravelTime` saves the travel time in the **Timing** attribute `tm` of the token for the statistical evaluation of the simulation results. With the arrival in the place **Waiting** the **Worker** token enters the submodel **MobileWork** (see Figure 3.13), which represents the process dependent part of the domain model. In this submodel the actual work is being performed in the transition **startWork** and the fusion place **Working**.

When a token is transferred into the fusion place **Working**, the worker-dependent working time for this task is added to the timestamp of the token by the function `W_workTime`. Additionally, the waiting time is saved by the function `W_adjustWaitingTime` for statistical evaluation. After the new timestamp of the token is reached by the simulator the token is transferred

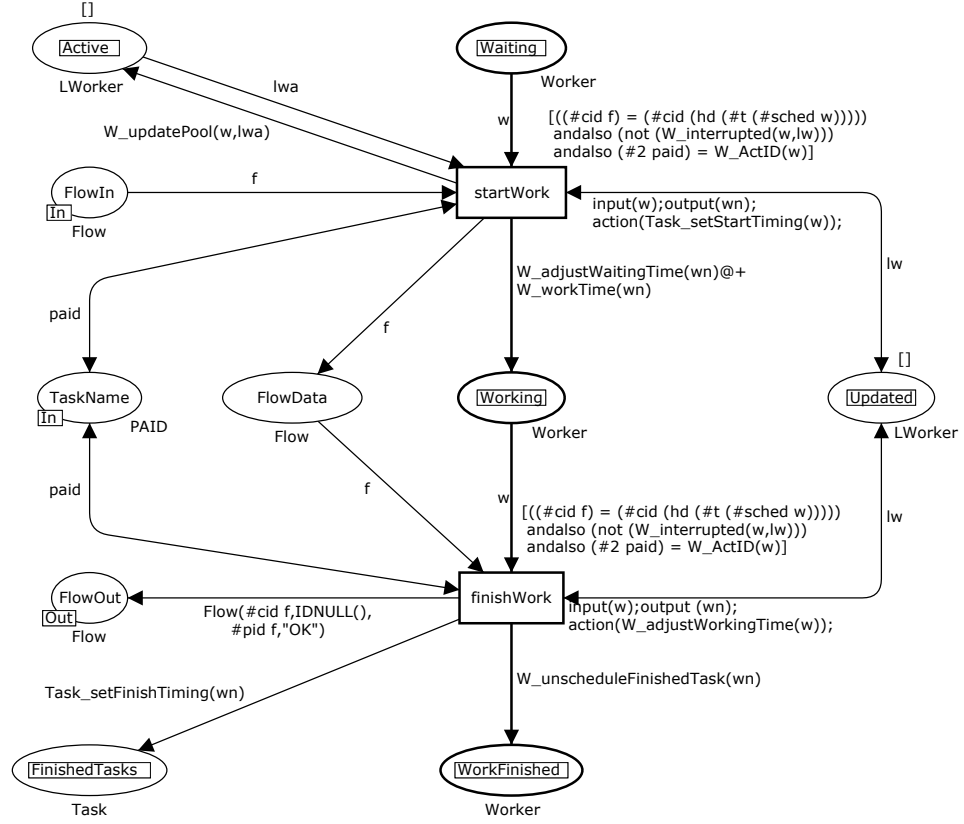


Figure 3.13: Model part MobileWork

to the transition `finishWork`. The action of this transition saves the working time for later evaluation by the function `W_adjustWorkingTime`. The currently finished task is removed from the schedule of the worker by the function `W_unscheduleFinishedTask`, and the token enters the fusion place `WorkFinished`. With the token reaching the place `WorkFinished`, the process dependent part of the resource state cycle is passed and the token shows up in the submodel `ResourceState` again.

In addition to the model elements, regarding the workers' state cycle the submodel `MobileWork` consists of model elements controlling the process flow and the overall organization of the work. The places `FlowIn`, `TaskName`, and `FlowOut` belong to the process control interface of the submodel and are described in Section 3.3.1. The fusion places `Active` and `Updated` are involved into the handling of urgent cases as already described above. The fusion place `FinishedTasks` saves all `Task` tokens after execution for later postprocessing.

As long as the schedule of the worker contains tasks to be performed, the token will be consumed by the transition `nextTask` and the state cycle is



started over with the next task to be executed. If the last task was executed (the length of the schedule is 0), the token is consumed by the transition `finishDay` indicating that the working day is over. The worker has then to travel home or to the depot of the region, respectively. As soon as all workers have reached the place `Home` the transition `toOpBase` is activated and the worker tokens are added to the `Inactive` pool and can be assigned with schedules for the next day by the submodel `Scheduler`.

### 3.2.7 Postprocessing the Simulation Results

All time related costs occur during the mobile activities of workers. Thus, the according time values are stored in the tokens of the workers as aggregated values of their working time, waiting time, and travel time. The non-aggregated time values are stored in the tasks by the function `Task_setFinishTiming` after the transition `finishWork` has fired. All finished tasks containing such aggregated information reside in the fusion place `FinishedTasks`. Cost related data is gathered in the submodel `GatherResults` which is depicted in Figure 3.14.

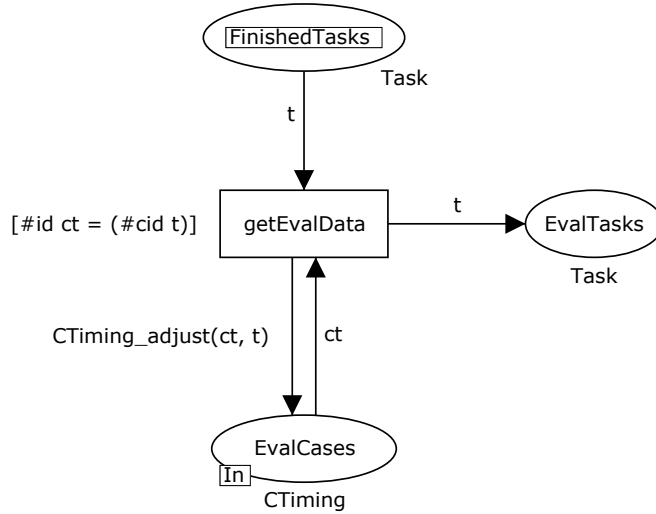


Figure 3.14: Model part `GatherResults`

The `CTiming` tokens in the port place `EvalCases` are initially created when the cases are generated in the submodel `Generator` (see Figure 3.9). The tokens of type `CTiming` contain cost related timing information of each finished case. Whenever a `Task` token enters the fusion place `FinishedTasks` in the submodel `MobileWork`, it becomes available to the transition `getEvalData`. This transition filters for the `CTiming` token corresponding to this task and adjusts the timing information of the case with the values of the task. Ad-

ditionally the `Task` token is stored in the place `EvalTasks` for later use. All gathered data is written to text files by CPN Tools after the simulation run.

### 3.3 Composition of Business Processes

Based on the model elements introduced in Section 3.2, this section handles the composition of mobile business processes utilizing the model elements. Business processes can be seen as an additional layer on top of the CPN domain model. This section demonstrates how this layer integrates with the CPN domain model by utilizing predefined fusion places and subpages. Further topics are the control structures necessary to create processes with sequences, branchings, and loops of activities.

#### 3.3.1 Process Modeling Interface

One of the main targets of this work is to separate process modeling from domain modeling. Nonetheless, the nature of CPNs and their execution in CPN Tools requires the process models and the domain model to be present in the same CPN model. Thus, both model types must be present in the same input file of CPN Tools. As we will see in Sections 3.4 and 5.4.2, the process models as the dynamic part of BPR projects will be developed in the SMPL language which is derived from UML ADs. The SMPL process models are transformed into CPN process models and interwoven with the CPN domain model of the preceding Section 3.2 in a single model file. Interweaving the models requires the domain model file to act as a template where the process models can be inserted easily. This is best supported if the domain model and the process models are kept separate and well defined insertion points are available in the template file. To keep the CPN domain model separate from the CPN process models as well as to establish well defined insertion points, an interface of the two was defined. The basic idea is to create a copy of the subpage `MobileWork` (see Figure 3.15) of the CPN domain model for every activity and to link these copies such that they form mobile business processes. As already introduced in Section 3.2, the subpage `MobileWork` contains all elements necessary to represent a mobile task.

To handle mobile business processes both the physical flow of the workers and the virtual control flow of the process have to be facilitated. For these reasons, the subpage `MobileWork` is equipped with two interfaces. The physical flow of the workers occurs through the places `Waiting` as input and `WorkFinished` as output. It is worth noting that these two places are fusion places which means that they are singletons. As a result, *all* waiting workers are present in *one and the same* place `Waiting`, regardless of the process and activity

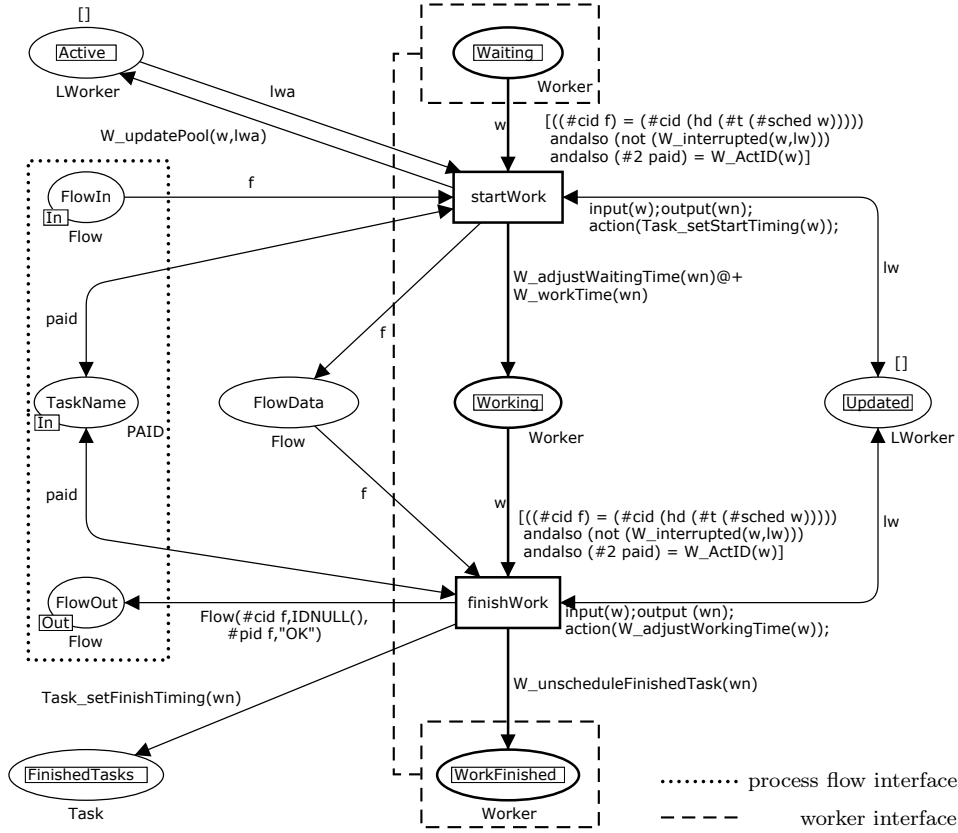


Figure 3.15: Interfaces of model part MobileWork

they are working on. The same applies for the workers present in the place `WorkFinished`.

The process control flow occurs through the places `FlowIn` as input and `FlowOut` as output. These places are of the type `Flow`. The third place of the process control interface is the place `TaskName`. This place contains exactly one token of the type `PAID` with the ID of the process and the ID of the activity this copy of `MobileWork` represents. This token is thus a unique identifier for the activity mapped to this subpage copy. The marking of the place `TaskName` is created at the creation time of the process model. Since the place builds loops with the transitions `startWork` and `finishWork`, its marking remains constant for the whole simulation run. In contrast to the worker interface, the process flow interface is built of ports instead of fusion places. Figure 3.15 gives an overview of the interfaces.

Structuring the interfaces as described above allows to integrate independent processes into the simulation as long as every process and every activity have a unique name which is represented by their respective attributes `ID` of type

**STRING**. The guard of the transition `startWork` of the worker interface ensures that a `Worker` token is always directed into the correct copy of the subpage `MobileWork`. This decision is taken based on a comparison of the name of the activity stored in both the schedule of the worker and the token `paid`. The process flow interface ensures that a worker cannot enter the subpage `MobileWork` until the respective activity is executable in the semantics of the business process. This is achieved by the input place `FlowIn` and the guard of `startWork` which blocks until a token is present in the place `FlowIn`.

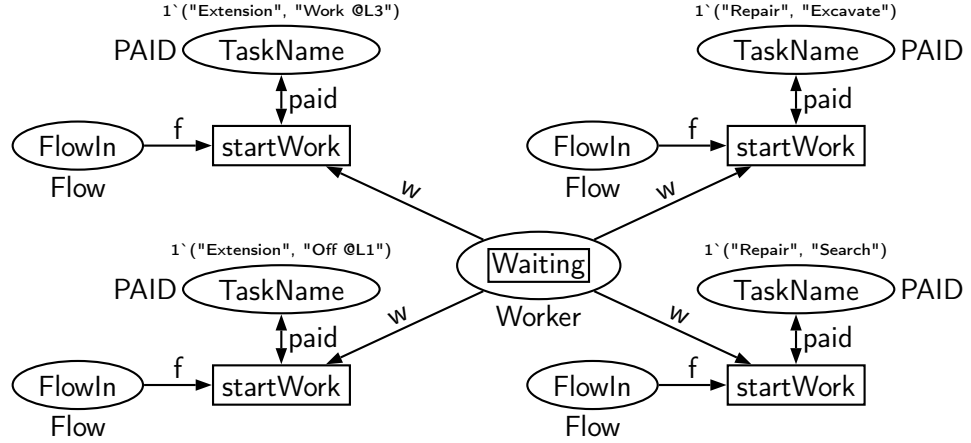


Figure 3.16: Process modeling interface instantiated

Figure 3.16 illustrates how the `MobileWork` interfaces are instantiated for enabling the transition `startWork`. The example in Figure 3.16 refers to the example processes introduced in Section 2.1 on page 13. Note that for the sake of lucidity Figure 3.16 is incomplete, since it shows only four activities instead of all ten activities of both processes.

Figure 3.16 shows a view of the interfaces that is internal to the subpage `MobileWork`. Since the business processes are created out of copies of `MobileWork` and utilize the interfaces, there is an external view to the interfaces too. The external view of the worker interface was already discussed in Section 3.2.6 in-depth.

The external view of the process flow interface is part of the actual business process descriptions. Since the CPN model of every activity is a copy of the subpage `MobileWork`, every activity needs a copy of the process flow interface, too. Figure 3.17 shows the CPN model substitution of an activity with its process flow interface. Note that in opposition to the internal view, the edge between the place `id` and the subpage `MobileWork` is unidirectional since the place `TaskName` is an input port.

The corresponding place names of the process flow interface are given in Table 3.1. Since all places must have a unique name, a unique positive integer

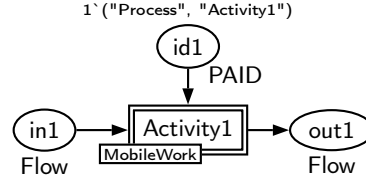


Figure 3.17: CPN substitution of a mobile activity

is appended to the names of the external interface places. For one and the same activity, the same unique number is appended.

Internal place	External place
FlowIn	in<number>
TaskName	id<number>
FlowOut	out<number>

Table 3.1: Corresponding places of the process flow interface

The **Flow** tokens will initially be created by the **Scheduler** as soon as the first task of a process is assigned to a worker's schedule. The tokens are created for all tasks with precedence 1. For tasks with higher precedence (in the power line extension example the activities "Work @L3", "On @L1", "On @L2") these tokens are produced as a result of the execution of the predecessor tasks of the same process—i.e., they represent the current state of the process. This happens on the edge from the transition **finishWork** to the place **FlowOut** after the execution of the task is finished. In the next sections it will be demonstrated how to build business processes out of the activity substitution depicted in Figure 3.17.

### 3.3.2 Control Structures

The set of control structures to be supported is delimited to (i) sequences, (ii) AND-branchings, and (iii) XOR-branchings. In [48] Gruhn and Laue evaluated 984 real world business process models including the SAP reference model [22; 61] and found that AND and XOR are the only necessary branching structures to model the vast bulk of business processes. For OR-branchings Gruhn and Laue found out that they can be replaced by AND and / or XOR in most cases. This is important since OR-branchings tend to cause semantic problems in business process models because their semantics is often misinterpreted as XOR by process modelers.

All control structures are defined such that their scope is local. This means that the model elements used to create a control structure have only knowledge about the process parts they are directly connected to. As an example

consider the power line damage search and repair process depicted in Figure 2.5 (p. 16). The control structure for the sequence of the activities “Excavate” and “Repair Cable” is local if it does not have any information about other activities or the XOR-branching of the process.

### 3.3.2.1 Process Part Connectors

Since Petri nets are bipartite graphs, two nodes of the same type must never be connected by a direct edge. To preserve the syntactically correct flow of control and data in the process model in certain situations, it is thus necessary to insert dummy nodes without function. The connector mechanism was already proposed by Staines in [112] as well as by Störrle and Hausmann in [113] for similar modeling situations. Figure 3.18 depicts the connectors.



Figure 3.18: CPN process part connectors

All connectors are named with the string `conn` followed by a unique positive integer (e.g., `conn2`). The naming is regardless of the connector’s type. Place connectors are always of the type `Flow`.

Note that the tool CPN Tools requires all model elements of a subpage to have a unique name. For this reason, numbers or other identifiers are sometimes appended to element names in the following. For illustration purposes see also Figure 3.18.

### 3.3.2.2 Sequences of Process Parts

A sequence of two or more activities can simply be modeled by connecting the subsequent activities with an intermediate transition connector. Figure 3.19 illustrates the result.

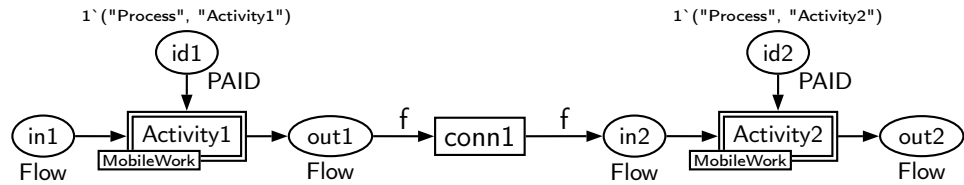


Figure 3.19: Sequence of activities in CPN models

Since the transition `conn1` has no function other than preserving the correct syntax of the process model and especially does not affect the model time, it does not influence the semantics of the process model. The inscription `f` of the connecting edges represents a token of the type `Flow` passing the control flow and the process state from the preceeding activity to the succeeding activity:

```
colset Flow = record cid: ID          (* ID of the Case *)
                      *   aid: ID      (* ID of the Activity *)
                      *   pid: ID      (* ID of the Process *)
                      *   state: STRING;
var f: Flow;
```

Note that `Activity2` is not activitated and can thus not be executed until a token of the type `Flow` is present in the place `in2`. For the sake of readability, in the remainder activities will be depicted as illustrated in Figure 3.20.

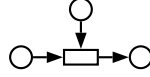


Figure 3.20: Substitution symbol of activities

### 3.3.2.3 Parallel Execution (AND)

Parallel execution of activities or compound process parts means that all branches of an AND-branching are executed and no order between activities of different branches is defined. Parallel execution is achieved by introducing a pair of transitions (`ANDSPLIT` and `ANDJOIN`), as illustrated in Figure 3.21.

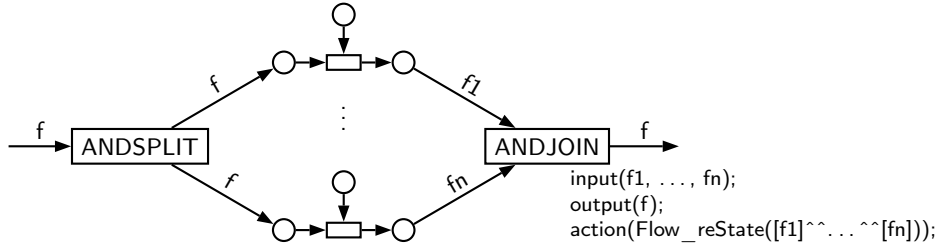


Figure 3.21: AND-branching in CPN processes

The transition `ANDSPLIT` copies the incoming `Flow` token such that the first activity in each branch can be activated (still depending on the appropriate `Worker` to be present in the place `Waiting` of the submodel `MobileWork`). The transition `ANDJOIN` collects the `Flow` tokens returned from each branch and calculates a new state as the result of the whole branching in the function `Flow_reState`. Note that the inscription of the transition `ANDJOIN` is a template. The dots represent an arbitrary number of additional branches

and incoming **Flow** tokens. If more than one AND-branching exists in the process, numbers must be appended to the transition names for uniqueness.

### 3.3.2.4 Alternative Execution (XOR)

The alternative or exclusive execution of activities or compound process parts means that exactly one branch of an XOR-branching is executed. An XOR-branching is achieved by introducing a set of transition pairs (**XORSPLIT** and **XORJOIN**) as illustrated in Figure 3.22. For each branch one pair of transitions is introduced. The **arb** (arbitrary) places of type **Flow** can either be input / output places of activities or connector places.

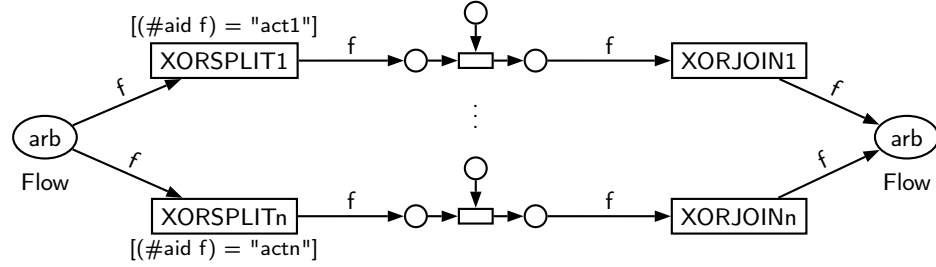


Figure 3.22: XOR-branching in CPN processes

The guards of the **XORSPLIT** transitions guarantee for the correct branch to be executed based on the activity's **ID** stored in the **Flow** token. If the branch to be executed has to be chosen randomly, the **XORSPLIT** transitions can be defined as illustrated in Figure 3.23. Since at least two split transitions and two join transitions are inserted per XOR-branching, numbers must be appended to their names for uniqueness.

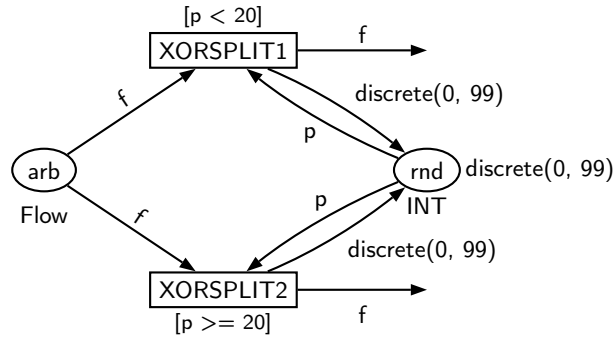


Figure 3.23: Random selection of the XOR-branch to be executed

The variable **p** is of type **INT** and the function **discrete** calculates discretely distributed random integers in the specified closed interval. The random



numbers are created both when the place `rnd` is initialized and whenever an `XORSPLIT` transition fires. Creating a random number with the firing of the respective `XORSPLIT` transition initializes the place `rnd` for the next execution of this process. In this example the transition `XORSPLIT1` fires with probability 0.2 and the transition `XORSPLIT2` fires with probability 0.8.

Note that if the workers are not scheduled dynamically (no urgent cases), XOR-branchings can simply be replaced by the activity or compound process part to be executed, because then the correct activities must be known to the scheduler before the planning period starts.

### 3.3.2.5 Repeated Execution (Loop)

A loop can be created of XOR-splits and XOR-joins. To keep the structural knowledge local to model elements, the guards of the XOR-splits are modified compared to a regular XOR-branching. See Figure 3.24 for illustration.

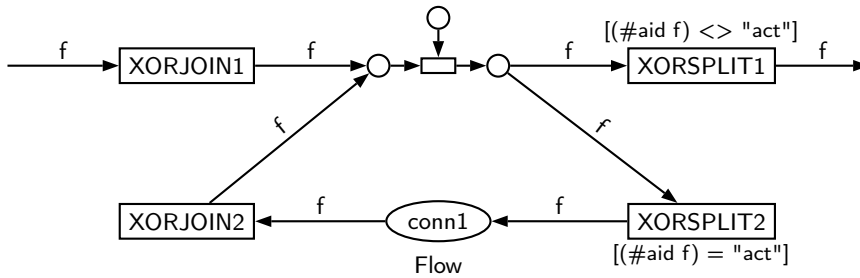


Figure 3.24: Loop in CPN processes

The loop control structure can also be constructed to handle the control flow randomized analog to XOR-branchings. This can be achieved by attaching the additional model elements and inscriptions described in Figure 3.23 to the transitions `XORSPLIT1` and `XORSPLIT2`.

Note that if the workers are not scheduled dynamically (no emergency reaction) loops can simply be replaced by a sequence of activities or compound process parts, because in this case the number of looped executions must be known to the scheduler before the planning period starts. The sequence contains as many copies of the activity or compound process part as the loop has to be executed.

### 3.3.3 Process Initialization

In Section 3.2.5 it is described how the subpage `Scheduler` creates `Flow` tokens for tasks with a precedence of 1 and stores them in the fusion place `InitialFlow`. The place `InitialFlow` must be a fusion place to keep the

domain model independent of the business processes. If there was a place for the initial flow tokens for every process, the scheduler would have to know these places. The fact that **InitialFlow** is a fusion place requires some attention when creating process models.

Every process model must contain the fusion place and must care for every **Flow** token to be present in the process flow interface it belongs to. As illustrated in Figure 3.25, this is achieved by testing the process ID and the activity ID of the token. The guard of the transition **init1** compares the appropriate attributes of the **Flow** token **f** in the fusion place **InitialFlow** to those of the **PAID** token **paid** in the interface place **id1** of the activity.

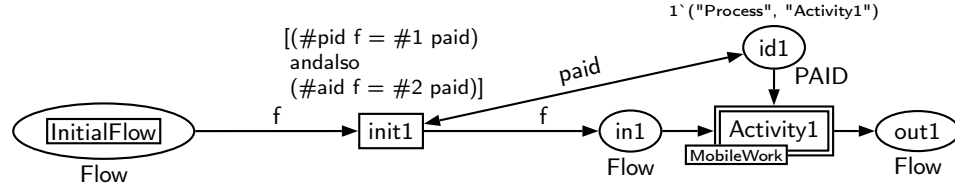


Figure 3.25: Initialization of a process

Since a process can contain more than one activity of precedence 1, the structure illustrated in Figure 3.25 has to be instantiated for each of these activities. The instantiation is regardless of the control structures before the activity, since tasks of precedence 1 have no preceding tasks and can always be executed as soon as the assigned worker is present at their location. Thus the control structures between the process entry point and the activities of precedence 1 can be omitted. Figure 3.26 shows the initialization structure for a process with  $n$  activities of precedence 1. For the sake of simplicity, the activities are substituted by the structure introduced in Figure 3.20. The **init** transitions are numbered to ensure unique names of all model elements.

### 3.3.4 Process Finalization

Since the **Flow** tokens are used to collect informations about the simulation run, they have to be stored for later evaluation. As for the initialization of the process flow in the scheduler, the collected data should be gathered independently from the processes being simulated. Thus, a fusion place **FinalFlow** of type **Flow** was created, which collects the flow tokens of all cases centrally. Figure 3.27 illustrates the structure for a process with exactly one activity having the maximum precedence of the process. The transition **conn1** has no business function but is just a connector between the two places.

If a branching join is the last element of a process model, it is possible, that more than one tasks have the maximum precedence of this process. In this

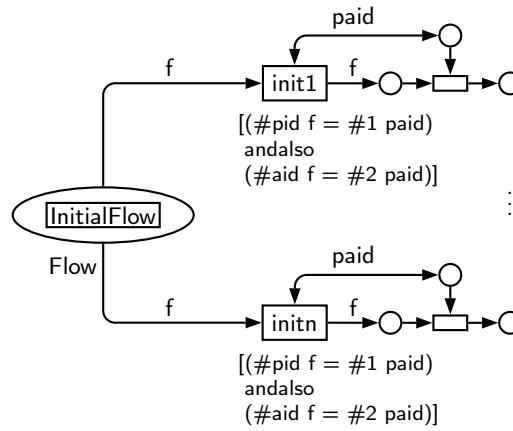


Figure 3.26: Initializing a process with multiple activities of precedence 1

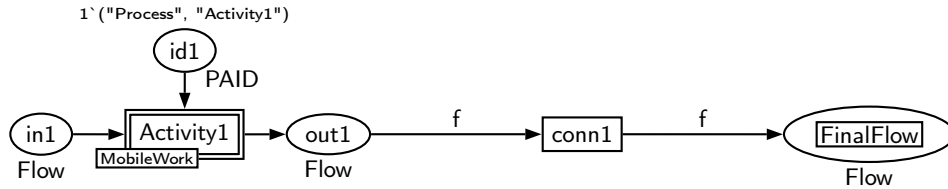


Figure 3.27: Finalization of a process

case, the Flow tokens are simply passed from the join's transition(s) to the fusion place `FinalFlow` without needing connector elements.

### 3.3.5 Section Summary

In this section we demonstrated how to compose business processes using the CPN domain model as a construction kit. Control structures for typical business process elements were introduced as well as modeling templates to initialize and to finalize business process models. Figures 3.28 and 3.29 show the CPN process models for the power line extension process and the power line damage search and repair process introduced in Section 2.1 (p. 13), which were constructed using this kit.

## 3.4 Reducing the Modeling Effort

Consulting Figures 3.28 and 3.29, it is easy to see that the CPN process models tend to be very complex and error-prone. This is due to the small set of nodes available, the interface copies, and the CPN inscriptions present in the process models. In addition, the substitution of process activities by

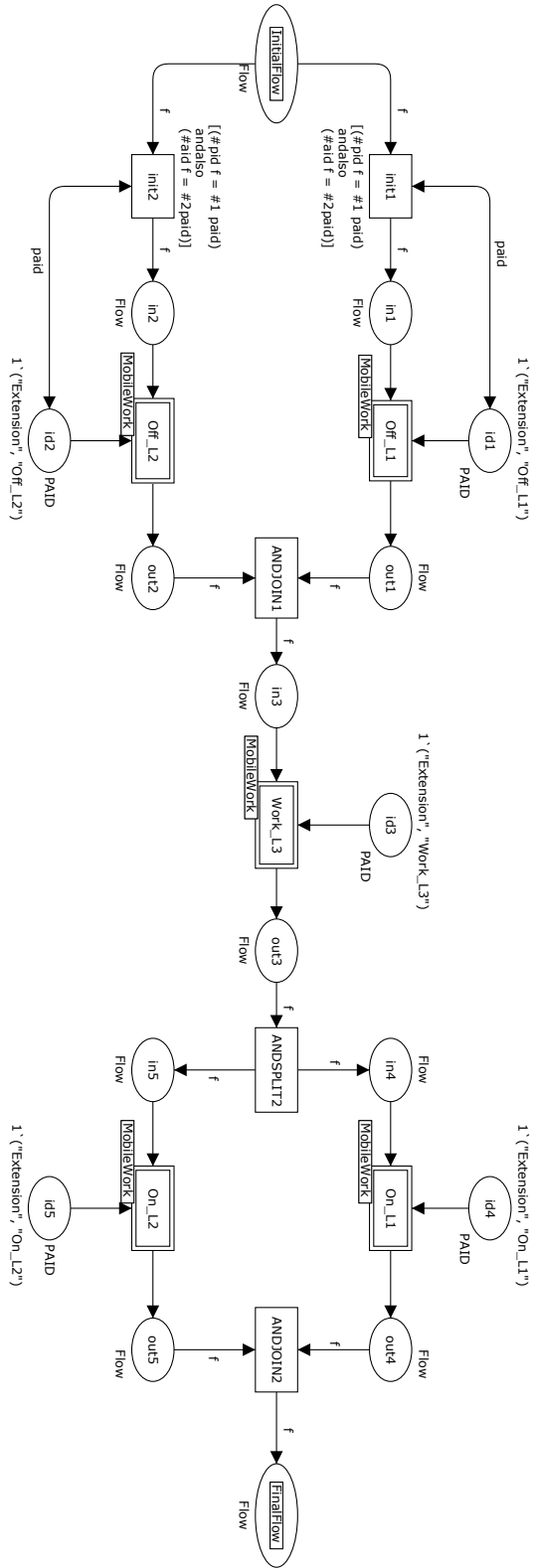


Figure 3.28: Resulting CPN process model for the utility example network extension process of Figure 2.3

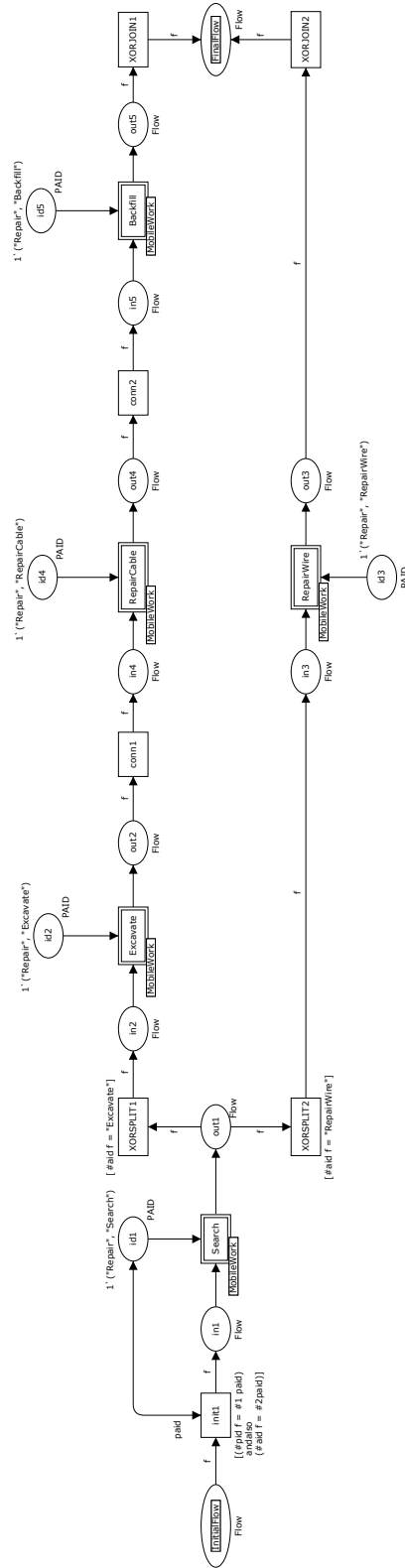


Figure 3.29: Resulting CPN process model for the utility example damage search and repair process of Figure 2.5

copying the subpage `MobileWork` leads to a very complex CPN of the whole organization far from comfortable manageability (remember that numerous business processes can exist in parallel and all of them become integrated into the same single CPN). Furthermore, CPNs are not commonly used as the modeling language of choice by business analysts.

It is thus necessary to offer such users the opportunity to model business processes in a more convenient and intuitive (semi-)formalism as, e.g., UML activity diagrams provide as well as to develop a methodology to transform these process models into simulatable Petri nets. Since Petri nets are the formal basis for the definition of UML activity diagrams, many approaches already define such transformation schemes, see [112]. Even though transformation schemes between activity diagrams and Petri nets exist, they do not reduce the complexity of the processes to model.

In [113] Störrle and Hausmann state that no intuitive mapping from UML activity diagrams into formal notations exists as, e.g., algebras or Calculus of Communicating Systems (CCS). In [112] Staines addresses the difficulty of handling native CPNs by transforming UML ADs into easy-to-read Fundamental Modeling Concept Petri Net Diagrams (FMC-PND) and then into CPNs which still does not offer reusability of generic models. Further approaches to transform UML diagrams into PNs are described in [57] (use cases of a banking system), [17; 74] (translation of UML ADs into stochastic PNs for performance analysis), [7] (formalization of the UML with Petri net semantics and higher level timed PNs), [35] (formalization of the UML based on CPNs), [108] (UML use case mapping to CPNs), and [31] (UML activity diagrams for workflow modeling). All of these approaches present methods that do not distinguish between reusable domain models and specific process models.

In this section a modeling language based on UML activity diagrams is introduced for the purpose of modeling mobile business processes. It supports the user to separate the reusable CPN domain model introduced in Section 3.2 from the specific CPN process models introduced in 3.3. This simple modeling language covers a subset of UML activity diagrams necessary for the control structures introduced in Section 3.3.2. A transformation scheme for the generation of CPN process models from activity diagrams is also introduced.

### 3.4.1 Modeling Language Requirements

The modeling language should cover the elements of processes as introduced in Section 3.3. It must not be bound or restricted to a certain business domain or a certain set of processes. It must be easy to learn and usable by business analysts. Other uses than the modeling of processes for the CPN

domain model are not intended and do not need to be supported. It is not necessary to model resources, locations, timing restrictions, or other elements of the domain model introduced in Chapter 2 since they are provided to the CPN through configuration files. Modeling execution probabilities is necessary only for alternative executions. The sum of the probabilities of all alternatives of one branching should always be 1.

The following list gives a summary of the requirements.

**REQ1** It must be possible to model control flow and its direction.

**REQ2** It must be possible to model activities.

**REQ3** It must be possible to model sequences of activities or submodels.

**REQ4** It must be possible to model parallel execution.

**REQ5** It must be possible to model alternative execution.

**REQ6** It must be possible to model repetitive execution.

**REQ7** It must be possible to model the execution probability of alternatives.

**REQ8** The language should be easy to use.

The question for compliance of a modeling language with the requirements REQ1 to REQ7 can be answered with “yes” or “no”, while the usability of a language (REQ8) is hard to measure. To gain rough estimations of the language’s usability, it is assumed that

1. the effort to learn a language increases with the size of the alphabet (number of element types), and
2. the risk of modeling failures increases with the number of elements necessary to model the structures described by the requirements REQ1 to REQ7.

With respect to these assumptions, the modeling language should provide an element type for each of the required structures and only few more. Well established languages for process modeling are Event-driven Process Chains (EPCs) [60], Petri Nets [92], Business Process Modeling Notation (BPMN) [46; 129], and UML activity diagrams [45]. Table 3.2 shows the properties of these languages with respect to the requirements.

It can be stated that the requirements REQ1 through REQ6 are met by all modeling languages without limitations. The requirements are either met by

Requirement	EPCs	Petri Nets	BPMN	UML AD
REQ1 Direction of control flow	Control Flow Arc	Arc	Sequence Flow	ControlFlow
REQ2 Activity	Function	Transition	Task	Action
REQ3 Sequence	Control Flow Arc – Event – Control Flow Arc	Arc – Place – Arc	Sequence Flow	ActivityEdge
REQ4 Parallel execution	2 AND connectors	possible, see [119]	Parallel Gateway	ForkNode and JoinNode
REQ5 Alternative execution	2 XOR connectors	possible, see [119]	Exclusive Gateway	DecisionNode and MergeNode
REQ6 Repetitive execution	2 XOR connectors	possible, see [119]	Sequence Flow Looping	MergeNode and DecisionNode
REQ7 Probability of alternatives	not possible	indirectly and in-exactly possible by Place – Transition cascades	indirectly possible by Conditional Sequence Flow	indirectly possible by Guards for ActivityEdges
REQ8 Number of element types	9	3	>40	>20

Table 3.2: Requirements match of process modeling languages



dedicated element types or by combinations of elements of different types. While EPCs and Petri nets consist of only few element types, their bipartite characteristics force the modeler to insert elements even if it seems not necessary for the actual situation to be modeled. As an example, consider the modeling of sequences (REQ2). UML activity diagrams provide dozens of element types [45]. Empirical evidence [44] shows that this comparatively large number seems to support optical differentiation, while the various ways to express facts promote modeling errors [59]. A similar estimation can be given for the BPMN. Due to the large number of element types both UML activity diagrams and BPMN are too voluminous for the modeling purposes of this work.

None of the languages evaluated directly supports modeling execution probabilities (REQ7). The cascading of places and transitions in Petri nets does not allow for the specification of a probability value but models the probability values in the cascading structure itself. BPMN allows to model constraints with the element type “Conditional Sequence Flow”. These constraints have a different semantics as probabilities but could be utilized for the purpose of modeling probability during the model transformation. Similar capabilities depending on evaluation during the model transformation exist in UML activity diagrams with the possibility to inscript “Guards” to outgoing edges of DecisionNodes.

Since none of the languages evaluated supports all requirements directly, it is necessary to develop a modeling language. This modeling language will be based on UML activity diagrams (UML AD). UML is a multi-purpose tool and business analysts usually work with it even when fulfilling tasks other than process modeling. It thus can be assumed that UML is widely known and understood by the target audience.

### 3.4.2 Simple Mobile Process Language Overview

The modeling language for mobile processes is named Simple Mobile Process Language (SMPL). It is closely related to UML ADs but reduced to just support the structures introduced in Section 3.3.2. Table 3.3 shows the elements of the SMPL and their UML AD counterparts. SMPL identifiers are chosen to minimize the differences to UML ADs.

The elements for the beginning and the end of a branch are modeled with the same symbol. The functions “start” and “end” of a branching symbol can be differentiated by the number of incoming and outgoing edges. A branching element with the function “start” has exactly one incoming edge and a branching element with the function “end” has exactly one outgoing edge.






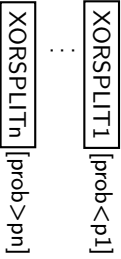

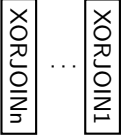



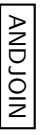

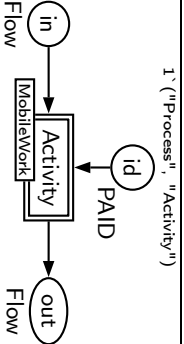


Function	SMPL type	UML type (section of spec. [45])	UML diagram element	CPN process model element
Process	Process	Activity		
Process start	InitialNode	InitialNode (12.3.31)		
Process end	FinalNode	ActivityFinalNode (12.3.6)		
XOR start	DecisionNode	DecisionNode (12.3.22)		
XOR end	MergeNode	MergeNode (12.3.36)		
AND start	ForkNode	ForkNode (12.3.30)		
AND end	JoinNode	JoinNode (12.3.34)		
Activity	Activity	Action (12.3.8)		
Control flow	ControlFlow	ControlFlow (12.3.19)		

Table 3.3: SMPL types and their counterparts in UML ADs and CPN processes

The grammar of the SMPL is given in Extended Backus–Naur Form (EBNF):

```

process      = 'InitialNode', block, 'FinalNode';

block        = 'ControlFlow', node, 'ControlFlow';

node         = node, block, node |
              andbranching |
              xorbranching |
              loop |
              'Activity';

andbranching = 'ForkNode',
              branch, branch, {branch},
              'JoinNode';

xorbranching = 'DecisionNode',
              branch, (branch | nop), {branch},
              'MergeNode';

loop         = 'beginloop',
              'MergeNode', block, 'DecisionNode',
              'endloop';

branch       = 'beginbranch', block, 'endbranch';

nop          = 'beginbranch', 'ControlFlow', 'endbranch';

```

Using this grammar, the power line extension process introduced in Section 2.1.2 can be expressed as follows:

InitialNode ControlFlow

ForkNode

beginbranch ControlFlow Activity ControlFlow endbranch

beginbranch ControlFlow Activity ControlFlow endbranch

JoinNode

ControlFlow Activity ControlFlow

ForkNode

beginbranch ControlFlow Activity ControlFlow endbranch

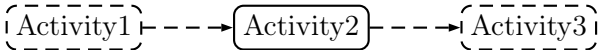
beginbranch ControlFlow Activity ControlFlow endbranch

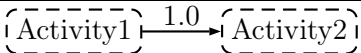
JoinNode

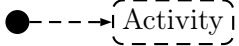
ControlFlow FinalNode

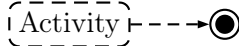
### 3.4.3 SMPL Elements and Semantics

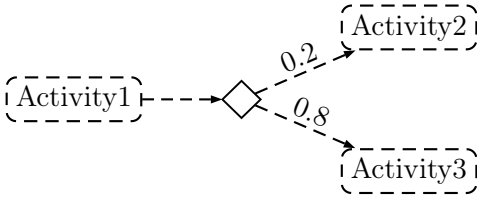
In the following, the element types of SMPL and their validity conditions are introduced. The graphical description uses example contexts. The element under consideration is depicted as given in Table 3.3, while the context elements are dashed.

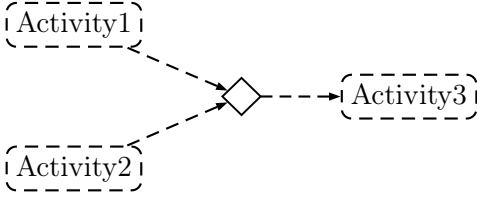
Name	Activity
Description	This model element represents an activity, as described in Section 2.3. The name of an Activity must be unique in the process it belongs to.
Predecessors	InitialNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
Successors	FinalNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
Example	 -

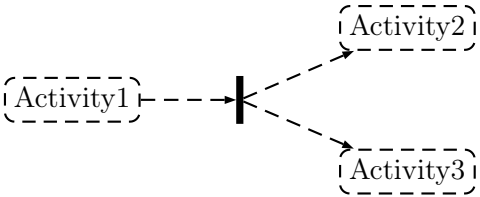
Name	ControlFlow
Description	The ControlFlow is the only edge type of the language. A ControlFlow has assigned a real probability value in the interval $]0, 1]$ . The probability value defaults to 1.0. Probability values different from 1.0 are only allowed on outgoing edges of DecisionNodes. For the sake of readability, probability values of 1.0 should not be drawn in process models.
Predecessors	—
Successors	—
Example	 -

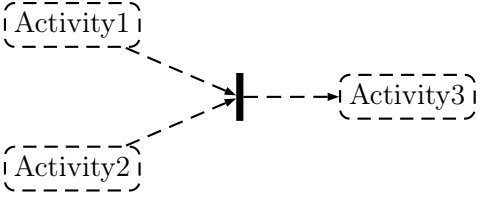
<b>Name</b>	<b>InitialNode</b>
<b>Description</b>	The InitialNode is the only node without predecessors. It represents the start of a mobile process. Per process exactly one InitialNode exists. This is ensured by the first rule of the grammar. All nodes reachable from a certain InitialNode belong to the same process.
<b>Predecessors</b>	none
<b>Successors</b>	DecisionNode, ForkNode, Activity
<b>Example</b>	 -

<b>Name</b>	<b>FinalNode</b>
<b>Description</b>	The FinalNode is the complement to the InitialNode. It is the only node without successors. It represents the end of a mobile process. Per process exactly one FinalNode exists. This is ensured by the first rule of the grammar. The flow of control ends at the FinalNode.
<b>Predecessors</b>	MergeNode, JoinNode, Activity
<b>Successors</b>	none
<b>Example</b>	 -

<b>Name</b>	<b>DecisionNode</b>
<b>Description</b>	The flow of control follows exactly one of several paths beginning at the DecisionNode. The path to be executed is chosen randomly based on the probabilities of the outgoing edges of the DecisionNode. Exactly one of the outgoing paths can be functionless, i.e., a DecisionNode can have zero or one MergeNode as successor.
<b>Predecessors</b>	InitialNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
<b>Successors</b>	DecisionNode, MergeNode, ForkNode, Activity
<b>Example</b>	 <p>-</p>

<b>Name</b>	<b>MergeNode</b>
<b>Description</b>	A MergeNode merges the paths of its corresponding DecisionNode.
<b>Predecessors</b>	DecisionNode, MergeNode, JoinNode, Activity
<b>Successors</b>	FinalNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
<b>Example</b>	 <p>-</p>

<b>Name</b>	<b>ForkNode</b>
<b>Description</b>	The flow of control follows <i>each</i> of several paths beginning at the ForkNode. Since a functionless path is useless in parallel execution, empty paths are not allowed.
<b>Predecessors</b>	InitialNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
<b>Successors</b>	DecisionNode, ForkNode, Activity
<b>Example</b>	

<b>Name</b>	<b>JoinNode</b>
<b>Description</b>	A JoinNode joins the paths of its corresponding ForkNode.
<b>Predecessors</b>	MergeNode, JoinNode, Activity
<b>Successors</b>	FinalNode, DecisionNode, MergeNode, ForkNode, JoinNode, Activity
<b>Example</b>	

SMPL meets the requirements given in Section 3.4.1. Since all but one element types of the language are identical to UML ADs, SMPL should be easy to use by business analysts. Nonetheless, there are mentionable differences to UML ADs:

- SMPL models have exactly one InitialNode and exactly one FinalNode.
- In SMPL an Activity represents an activity as introduced in Section 2.3, which is equivalent to UML AD Actions.
- In SMPL it is not possible to specify preconditions and postconditions of Activities as it is possible for UML AD Actions.
- In SMPL ControlFlows can be assigned with a probability value. This is similar but not equivalent to UML Guards.

- In SMPL branching nodes (DecisionNode, MergeNode, ForkNode, JoinNode) must always exist as pairs.
- In contrast to UML ADs, a branching node has always either exactly one incoming edge or exactly one outgoing edge.

### 3.4.4 Transformations of SMPL Models to the CPN Domain Model

By utilizing the general substitution scheme introduced in Figure 3.17 and the process structures of Section 3.3.2, a general set of transformations from SMPL diagram elements to CPN process model elements (see Table 3.3) can be derived. A preliminary version of the transformation has already been published by the author in [50]. Since processes consist of combinations of SMPL model elements, the respective transformations to CPN process model elements are given in Figures 3.30 to 3.43, where the left side (a) shows the SMPL model and the right side (b) shows the resulting CPN. The basic idea of the transformations was already introduced in Section 3.3.2. Note that the transformations are executed in the order of the control flow, beginning with the InitialNode, ending with the FinalNode of the process.

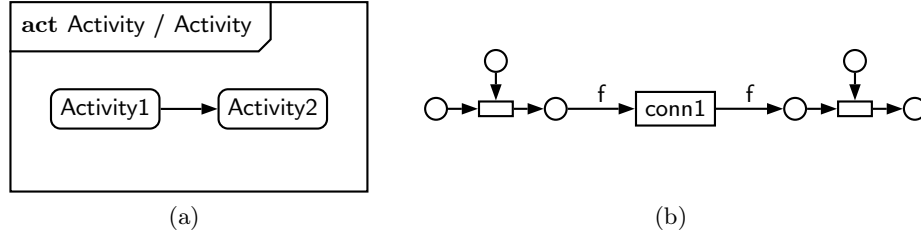


Figure 3.30: Composition of a sequence of two Activities

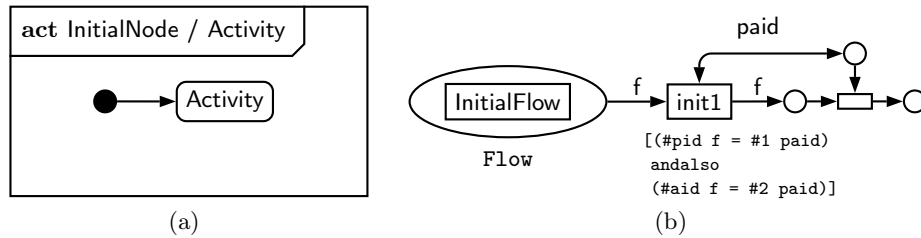


Figure 3.31: Composition of an InitialNode and an Activity



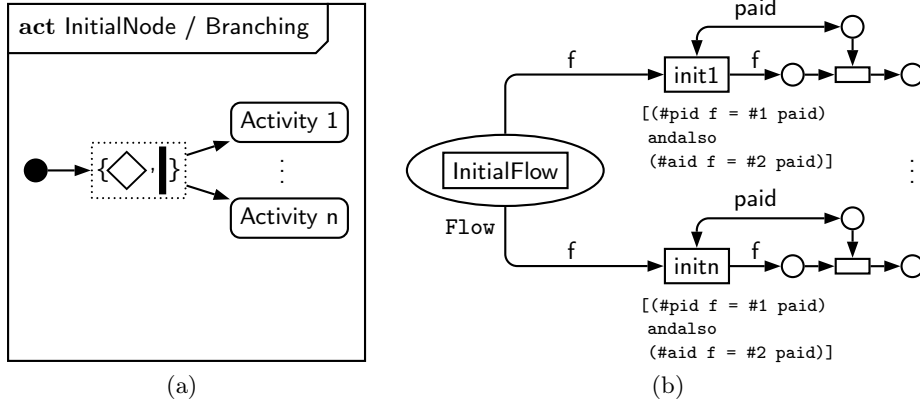


Figure 3.32: Composition of an InitialNode, Branchings, and Activities

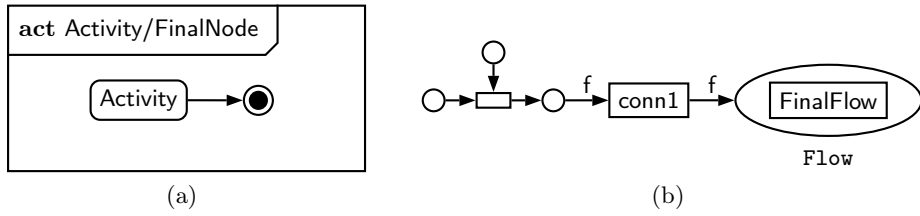


Figure 3.33: Composition of an Activity and a FinalNode

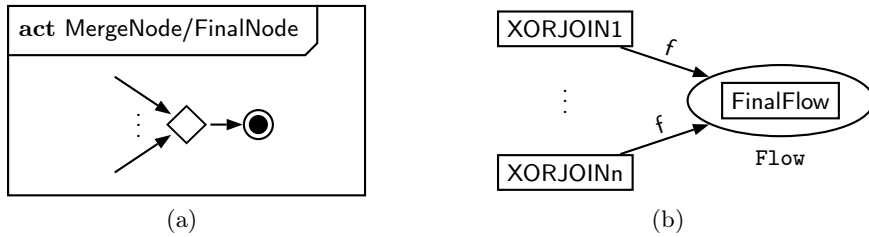


Figure 3.34: Composition of a MergeNode and a FinalNode

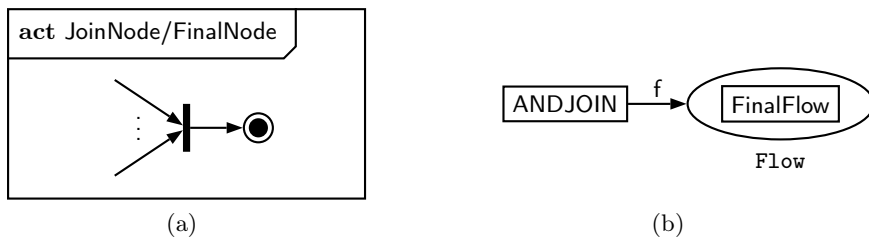


Figure 3.35: Composition of a JoinNode and a FinalNode

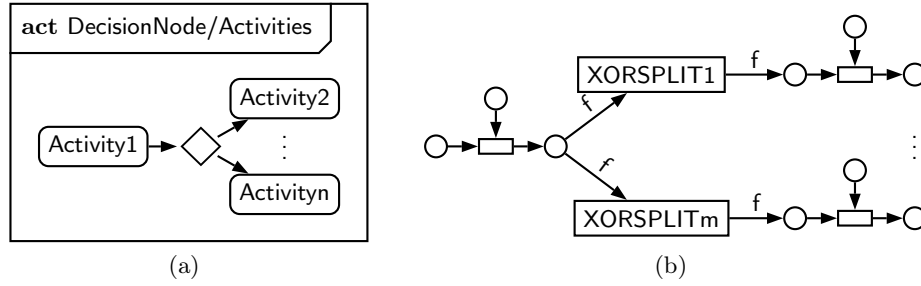


Figure 3.36: Composition of Activities with an intermediate DecisionNode

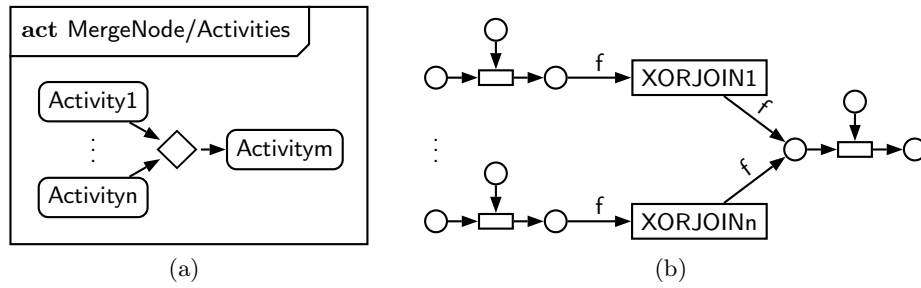


Figure 3.37: Composition of Activities with an intermediate MergeNode

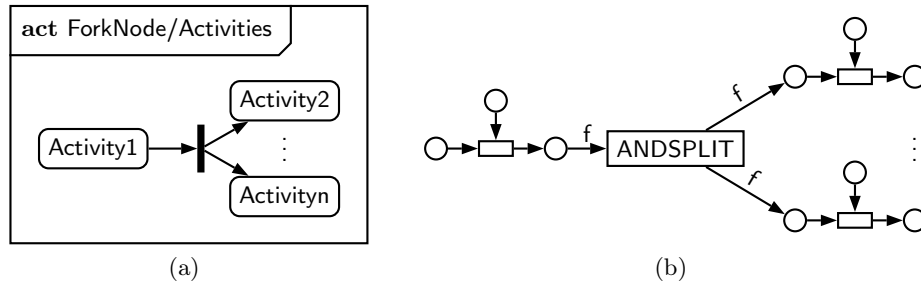


Figure 3.38: Composition of Activities with an intermediate ForkNode

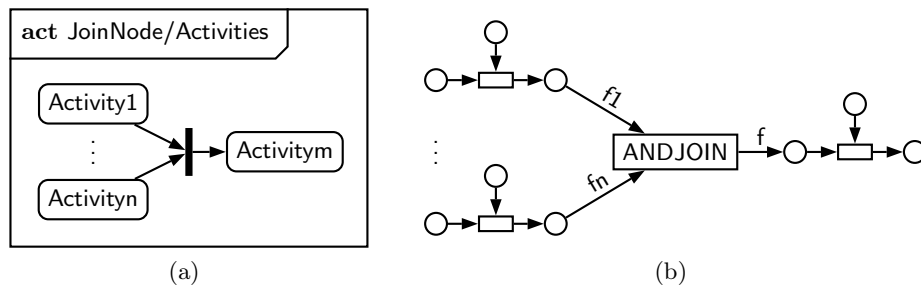


Figure 3.39: Composition of Activities with an intermediate JoinNode

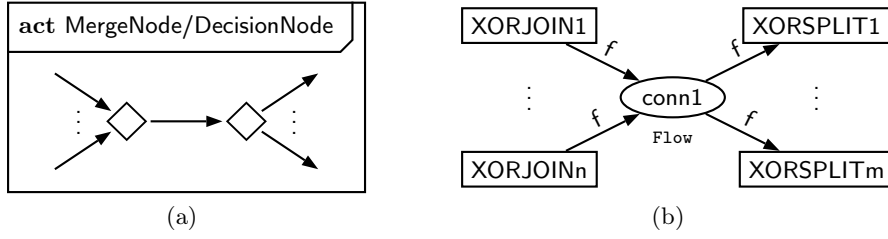


Figure 3.40: Composition of a MergeNode and a DecisionNode

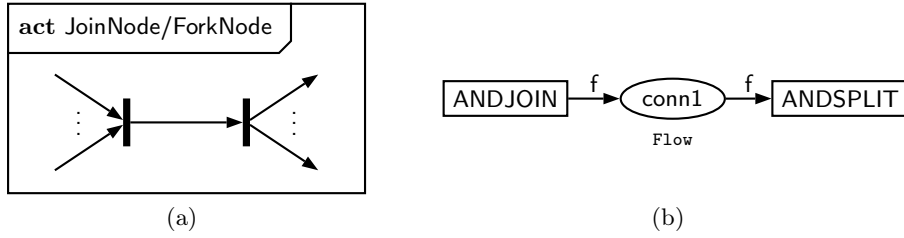


Figure 3.41: Composition of a JoinNode and a ForkNode

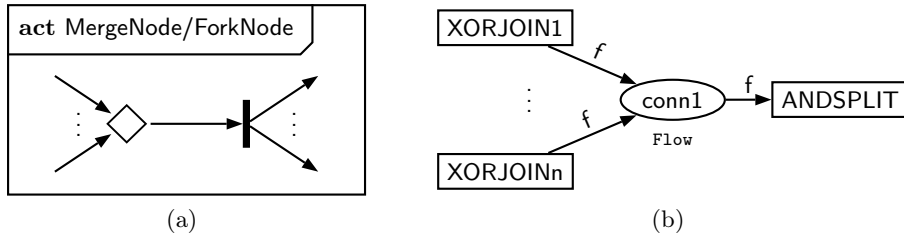


Figure 3.42: Composition of a MergeNode and a ForkNode

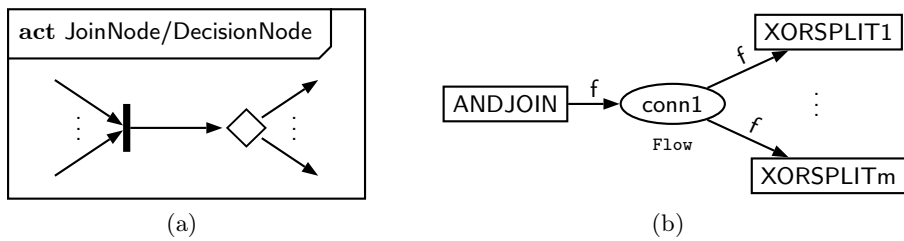


Figure 3.43: Composition of a JoinNode and a DecisionNode

### 3.5 Chapter Summary

In this chapter a domain independent model of mobile work was introduced. The model can be utilized to simulate the mobile operation of an enterprise in a way that incorporates all mobile business processes into the same simulation. The model supports the separation of concerns such that business specific process models can be developed independently from the special properties of mobile work. Business users can model their specific processes with the Simple Mobile Process Language—a subset of UML activity diagrams—and utilize the execution power of colored Petri nets without understanding them in-depth. This is achieved by introducing transformation patterns from SMPL to colored Petri nets and an interface definition between the process models and predefined domain models. The resulting executable process models inherit all constraints of the underlying domain “Mobile Work” from the domain model without the need for the business analyst to model these constraints explicitly. The approach itself is not restricted to the domain of mobile work but introduces a general way to define arbitrary domain models and to use them for the composition of executable process models by the definition of an appropriate interface to the domain model. A further advantage of the approach is the reusability of the domain models and, thus, the shifting of modeling complexity from repeated process modeling projects to centrally controllable singular domain modeling.

## Chapter 4

# Mobile Workforce Scheduling

In this chapter the aspects of workforce scheduling in general and for mobile organizations in particular are discussed. Scheduling and sequencing are concerned with the optimal allocation of scarce resources over time. Scheduling deals with defining which activities are to be performed at a particular time. Sequencing concerns the order in which the activities have to be performed. The allocation of scarce resources over time has been the subject of extensive research since the early days of operations research in the mid 1950s. Scheduling and sequencing theory is characterized by a virtually unlimited number of problem types [54].

We will demonstrate that the scheduling of mobile workers is a generalization of the Vehicle Routing Problem (VRP) [115] and the Resource-Constrained Project Scheduling Problem (RCPSP) [66]. Based on the optimization objectives introduced in Section 2.5, the Mobile Workforce Scheduling Problem with Multitask-Processes will be introduced and formalized. To support finding solutions of the MWSP-MP, the necessary neighborhood operators for inserting and removing cases as well as tasks are described in-depth. Furthermore, three policies to acquire an initial solution are introduced.

### 4.1 Introduction

According to the optimization objectives introduced in Chapter 2 the major aim of workforce scheduling is to reduce the overall process execution costs. These costs are mainly influenced by the importance (priority) attached to the cases, the amount of workers' travel, the spatial distribution of the workforce over time, the utilization of the workforce, and the violation of cases' time windows. As already stated, a mobile workforce operates in environments where the availability of a wireless data connection is uncertain, and

the state of workers may thus be unknown. As a result scheduling urgent cases may not be possible at minimal costs as will be shown in this chapter.

During a standard planning period (e.g. a working day), a mobile worker performs mobile tasks of different kind that match his qualifications. The different tasks of a worker's daily schedule may belong to one case or to different cases, depending on the cost-driven workforce scheduling. The scheduler aims at assigning tasks to workers such that the overall costs of the process execution become minimal. This means to consider cases and tasks present in the system at planning time—usually before the start of each working day—and to consider cases and tasks emerging during the course of the working day. Cases that emerge during the working day may only be considered for spontaneous scheduling if their execution is urgent—i.e., the time window demands the execution on the same day. This work differentiates between *planned* scheduling, which results in the initial schedules for the workforce for the whole working day, and *dynamic* scheduling, which modifies the initial schedules during the working day with respect to the current situation of the business operation.

To illustrate the special challenges of mobile environments in terms of workforce scheduling, the scheduling problem will be introduced with examples of both non-mobile business processes and mobile business processes.

## 4.2 Scheduling Paradigms

As this work deals with mobile environments in which the normal operation of the workforce may be interrupted by emergency situations demanding immediate reaction, the scheduling of the workforce must support different types of work assignment. This includes both creating an initial daily schedule for each worker and adapting existing schedules for the support of urgent cases. The following definitions distinguish the respective terms used in the next sections.

**Definition 4.1 (Urgent case)** *A mobile case is called urgent if it emerges after the schedules of the workers have already been created for the planning period and the business requirements demand to execute that case during the same planning period.*

**Definition 4.2 (Standard case)** *A standard case is a mobile case that is not urgent.*

**Definition 4.3 (Planned scheduling)** *Planned Scheduling is the assignment of standard cases to workers' schedules before the start of the planning*

period. The resulting schedules cover the whole planning period. The scheduling is finished before the planning period starts. The scheduling must not start before the end of the preceding planning period (see Definition 2.14).

#### 4.2.1 Planned Workforce Scheduling

##### 4.2.1.1 Non-mobile Business Processes

Executing non-mobile tasks requires planning as soon as either tasks with time windows or more tasks than adequately skilled workers exist. Depending on the properties of the tasks the planning period may be the whole day or only a couple of hours. The following example will illustrate the coherencies.

Three mutually independent cases of simple processes are given:  $c_1, c_2$ , and  $c_3$  (see Section 2.1 for the definition of such processes). Each of them consists of a single task, named  $\tau_1, \tau_2$ , and  $\tau_3$ . Additionally, a task  $\tau_b$  is introduced which indicates that each worker has to take a rest of 30 minutes between 11:00h and 14:00h. The properties of the tasks are illustrated in Table 4.1.

Task	$t_a^{def}$ in hrs	$t_\tau^{min}$	$t_\tau^{max}$	$\kappa_c^{uv}$ per hr
$\tau_1$	1.0	08:00h	10:00h	100
$\tau_2$	2.0	08:00h	16:00h	100
$\tau_3$	3.0	08:00h	15:00h	100
$\tau_b$	0.5	11:00h	14:00h	100

Table 4.1: Properties of example tasks

In this example the time unit is considered to be one hour and the cost per time unit  $\kappa_w^u$  of a standard worker is supposed to be 30. This is the real value of the power and gas supply described in Chapter 2. To accomplish the tasks, one worker,  $w_1$ , is available.

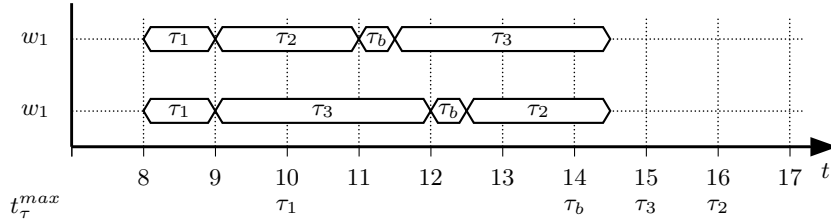


Figure 4.1: Assignment of non-mobile tasks

To comply with the time windows, the tasks can be executed either in the order  $\tau_1, \tau_2, \tau_b, \tau_3$  or  $\tau_1, \tau_3, \tau_b, \tau_2$  (see Figure 4.1 for illustration). Thus, the

scheduler has to plan the task  $\tau_1$  to be the first task before the working day begins. The further order can remain undecided until 10:00h. The total execution costs are just the worker related costs for the time between 08:00h and 14:30h which aggregate to 195.

Though the tasks  $\tau_1, \tau_2$ , and  $\tau_3$  belong to different cases and have thus no business relation, their co-existence in the same system induces interdependencies between them. With the conditions mentioned above the tasks  $\tau_2$  and  $\tau_3$  must be executed after  $\tau_1$  though no business requirements demand this order. Thus, work cannot be assigned freely, if resources are rare. As a result, tasks of actually independent business processes may become interdependent.

#### 4.2.1.2 Mobile Business Processes

When planning mobile tasks, not only the availability of resources and the time windows of tasks and cases have to be considered but additional challenges as, e.g., the location dependency of tasks and the lacking trustworthiness of wireless data connections, must be considered as well. The former challenge leads to travel efforts and thus to time gaps between the different tasks of one worker while the latter challenge forces the creation of daily schedules for the workers. Due to the need of minimizing the total execution time of certain cases, it is further necessary to synchronize the schedules of different workers such that the execution of subsequent tasks can take place seamlessly with only small or even without time gaps. Thus, the travel times have to be considered as part of the schedules and appropriate calculations or approximations must be applied. Based on the considerations of the optimization objectives mentioned in Section 2.5, it further seems legitimate to assume that reducing travel efforts leads to lower total execution costs. Thus, the tasks assigned to one worker should probably be located closely together.

To transfer the example introduced in Section 4.2.1.1 to a mobile environment, it is extended as follows. Figure 4.2a shows the locations A, B, and C as the execution sites of the tasks  $\tau_1, \tau_2$ , and  $\tau_3$  as well as the expected travel times between the three locations.

The execution times, time windows, and costs of the previous example are retained. Additional costs arise from the workers' traveling. The travel costs per distance unit are considered to be 0.3 per kilometer. It is further considered that the average speed is 50 kph so that the costs arising from the travel distance can be transformed into additional costs per travel time unit and aggregate to 15 per hour.

Considering the travel times stated in Figure 4.2a the only possible execution order with all time windows met is  $\tau_1, \tau_3, \tau_b, \tau_2$  (see Figure 4.2b, bottom).



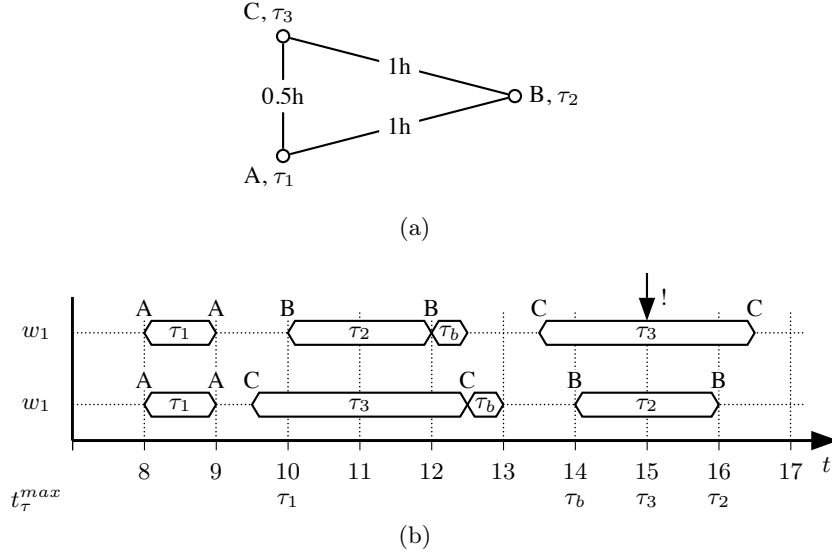


Figure 4.2: Assignment of mobile tasks

This order can be performed at a total cost of 262.5 (240 worker's time + 22.5 travel). It is further necessary to ensure the start of the task  $\tau_1$  at the location A at 08:00h. The break can take place anywhere between the locations C and B as long as it does not cause deviations and additional travel time. The alternative execution order  $\tau_1, \tau_2, \tau_b, \tau_3$  leads to the violation of the time window of  $\tau_3$  at 15:00h and thus to the generation of additional costs (see Figure 4.2b, top). This execution order would lead to total costs of 435 (255 worker's time + 30 travel + 150 violation of the time window).

In this example the restrictions of mobile environments as introduced in Chapter 1 apply. Since network connections are unreliable it must be considered that schedules cannot be updated during the working day. Thus, the schedule of the worker  $w_1$  has to be complete for the whole day before he travels to the location A. This schedule does not give leeway for changes. Even the possible availability of a data connection at one of the visited locations does not loosen these restrictions. As this example handles just one worker and four tasks, it is reasonable to assume that a significantly larger number of tasks and workers and the demand for different qualifications may lead to great difficulties when creating the daily schedules for the workers.

#### 4.2.2 Dynamic Workforce Scheduling

In addition to planning schedules before the planning period starts, several business situations can demand the change of schedules due to the emergence

of urgent cases.

Urgent cases can, for instance, be emergency situations as illustrated by the damage search and repair process in Section 2.1.3 (p. 15). Another example for such business cases are short-term service agreements with high contractual penalties. To avoid highly increasing execution costs of the cases, it is necessary to assign these cases immediately to appropriate workers—eventually postponing already assigned or even active cases.

**Definition 4.4 (Dynamic scheduling)** *Dynamic scheduling is the assignment of an urgent case after the beginning of the planning period.*

The operation of the scheduler in such situations can either be controlled by setting (i) the priority of the case or (ii) the time window of the case and the according time window violation costs. To schedule urgent cases, it is possible that a sufficient number of adequately skilled workers is either available or not available. If the former situation applies, the scheduler may simply assign the tasks of the case to the workers while the latter situation usually requires comprehensive treatment. Nonetheless, the scheduler should usually not just assign available workers without considering the overall costs resulting from its decision.

#### 4.2.2.1 Non-mobile Business Processes

Figure 4.3 shows an extension of the example introduced in Section 4.2.1. At 09:30h a new task— $\tau_n$ —with the time window [9:30h, 12:00h] and a default execution time of one hour emerges.

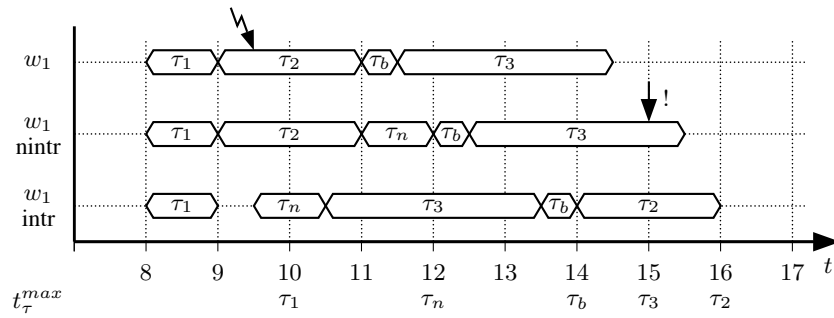


Figure 4.3: Dynamic scheduling of non-mobile tasks

The first sequence of Figure 4.3 shows the schedule of the worker  $w_1$  as planned for the working day. The second sequence of 4.3 shows the situation if the worker  $w_1$  finishes his current task before he executes the new task. The original order of the tasks in the schedule is retained but the time window

of the task  $\tau_3$  is violated by 30 minutes such that the total execution cost aggregates to 275. By contrast, the third sequence of Figure 4.3 is created by the scheduler instructing the worker  $w_1$  to interrupt the execution of his current task  $\tau_2$ . The worker immediately begins to work on the new task  $\tau_n$ . Note that the order of  $\tau_2$ ,  $\tau_3$ , and  $\tau_b$  is completely different than in the original schedule of the worker. With the interrupt and postponement of  $\tau_2$ , all tasks can be executed without violations of their respective time windows at a total cost of 240.

In the situation illustrated in Figure 4.3 it is cheaper to interrupt the task  $\tau_2$ . In different situations it can be cheaper to complete  $\tau_2$ . Depending on the expected costs, the scheduler has to decide how to continue the working day which may imply to overrule and redistribute a large number of schedules.

#### 4.2.2.2 Mobile Business Processes

In mobile environments it may be necessary to assign tasks dynamically in situations as the damage search and repair process illustrated in Figure 2.5 (p. 16). If a damage occurs during normal working hours, the workers are usually busy with tasks in the field. The sudden breakdown of the power supply of an area as a result of the damage requires preferred attention and treatment.

The implications of such a situation for the example introduced in the preceding Sections are illustrated by Figure 4.4. At 09:30h a new task— $\tau_n$ —is ready for execution. Referring to the damage search and repair process, this could mean that the damage was perceived and the location to start the damage search has been identified. The new task has a time window [9:30h, 12:00h] and a default execution time of one hour. The location of the task  $\tau_n$  is D, as illustrated in Figure 4.4a. The depot of the organization is situated at location A. Note that location D can be reached from A within 90 minutes so that the time window of  $\tau_n$  would not be violated if the travel from A to D starts immediately.

As for the non-mobile business processes, we will examine the implications of task interruption and task continuation for the mobile processes. The first sequence of Figure 4.4b shows the schedule of the worker  $w_1$  as planned for the working day. If the worker  $w_1$  is not assigned to perform the task  $\tau_n$ , this schedule will be executed by  $w_1$  as planned. The third sequence shows the schedule of the worker  $w_1$  if he is instructed to abandon his current schedule and to perform the task  $\tau_n$  immediately.

In both cases it is not possible to schedule the tasks such that worker  $w_1$  can perform all tasks without time window violations or just within his standard working day. It is thus necessary to involve a second worker,  $w_2$ ,

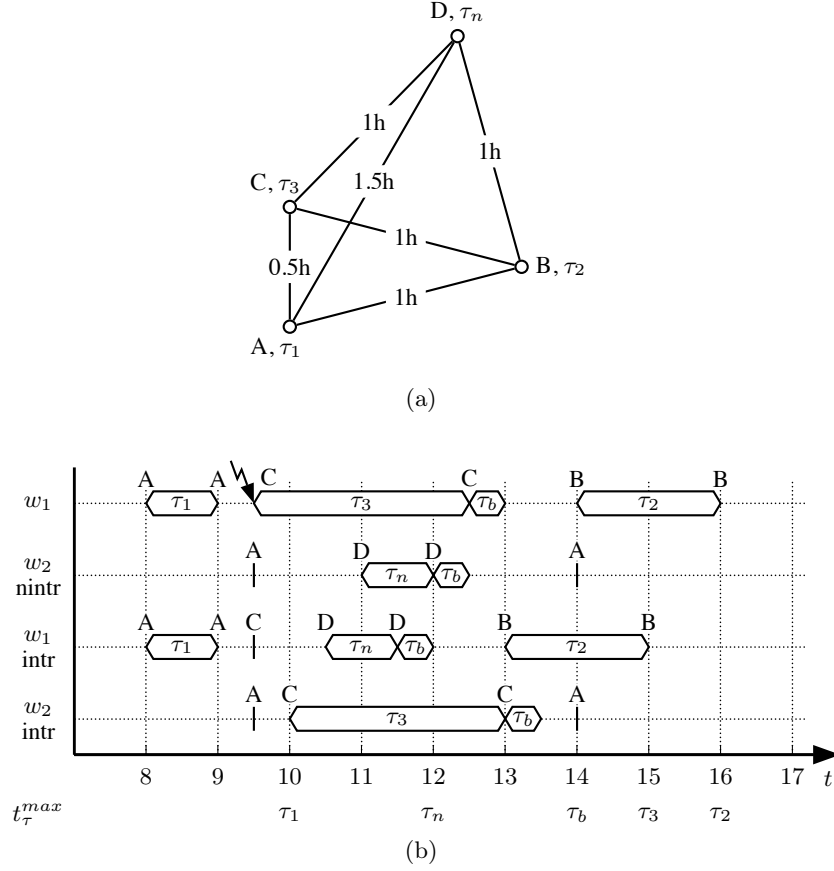


Figure 4.4: Dynamic scheduling of mobile tasks

in the execution of the task  $\tau_n$ . Worker  $w_2$  can be involved either directly or indirectly. The former indicates that  $w_2$  travels to location D and executes the task  $\tau_n$  as illustrated in the second sequence of Figure 4.4b. The latter, by contrast, indicates that the schedule of  $w_1$  is altered to execute  $\tau_n$  and one of the tasks originally assigned to  $w_1$  is assigned to  $w_2$ , as illustrated in the fourth sequence of Figure 4.4b. In this scenario it is assumed that worker  $w_2$  is currently situated at the depot and is available for immediate assignment. If no worker is currently available, it is also possible to abandon a whole case for the current working day to gain flexibility for the assignment of an urgent case, which would be treated in a similar way. This could involve a large number of workers and could—as the extreme consequence—lead to the rearrangement of the schedules of all workers.

If the schedule of worker  $w_1$  is not altered and  $w_2$  performs the task  $\tau_n$ , the total costs aggregate to a value of 442.5 (240 + 135 workers' time + 22.5 + 45 travel). If the schedule of  $w_1$  is altered and  $w_1$  performs the task  $\tau_n$ , the total costs aggregate to a value of 397.5 (210 + 135 workers' time + 37.5

+ 15 travel). It is thus cheaper to instruct the worker  $w_1$  to interrupt the execution of  $\tau_3$ —actually instruct  $w_1$  not to start  $t_3$ —and to assign  $\tau_n$  to  $w_1$  and further to assign the now unassigned task  $\tau_3$  to the worker  $w_2$ .

The scheduling of mobile cases may be aggravated further by the fact that the tasks to be performed may not be known until parts of a case have already been executed and finished. For illustration purposes consider the task  $\tau_n$  of the example of Figure 4.4 to be the activity *Search* of the damage search and repair process introduced in Section 2.1.3.

It is then possible that the case has to be scheduled in two portions, since it may be unclear in which part of the network the damage occurred. Accordingly, the repair part of the case may be planned after the search is finished, since the repair of a cable demands different skills and equipment than the repair of a wire.

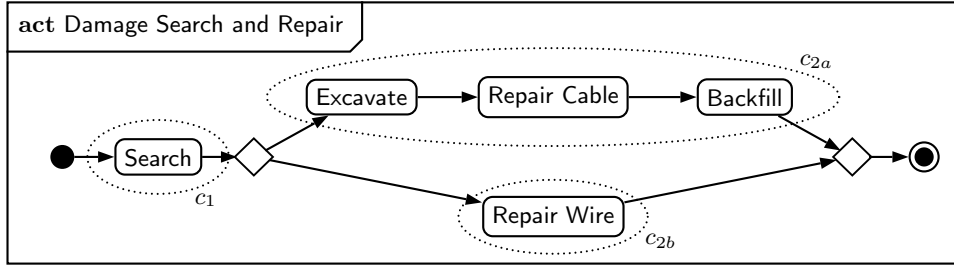


Figure 4.5: Power line damage search and repair process - splitting of a case

To sustain the method of scheduling only complete cases, the case is split into two different sub-cases  $c_1$  and  $c_2$  as illustrated in Figure 4.5. The sub-case  $c_1$  is an instance of the simple process *Damage Search*. For the sub-case  $c_2$ , it remains unclear which tasks have to be performed until the search in sub-case  $c_1$  is finished. Thus, the second sub-case is either  $c_{2a}$  *Repair Cable* or  $c_{2b}$  *Repair Wire*. If a case has to be split into different portions, the timing restrictions can always be managed with appropriate time windows for the sub-cases and their respective tasks.

It could be argued, that dynamic scheduling can be achieved by simply performing a new planned scheduling run. A scheduler run for dynamic scheduling, however, has to consider many properties of the situation which do not apply for planned scheduling. Among these properties are the (possibly unknown) state of active cases and the probably necessary rollback of already performed tasks, which has to be scheduled, too. Thus, dynamic scheduling requires advanced algorithms compared to planned scheduling.

### 4.3 Related Scheduling Problems

Scheduling the mobile workforce is closely related to well known problems of the operations research domain. In the case where each process consists of exactly one task, the problem can be interpreted as a variant of the Vehicle Routing Problem with Time Windows (VRPTW). Comprehensive surveys on the VRP and the VRPTW are given in [71] and [20]. Different skills and qualifications of workers can be interpreted as order/vehicle compatibility constraints which are commonly used for the Heterogeneous Fleet VRP, which recently was surveyed by [6]. There are several generalizations of the VRP in which each transportation request consists of more than one task. In the Pickup and Delivery Problem (PDP) [88; 89] each transportation request consists of exactly one pickup task and one delivery task. In the General Pickup and Delivery Problem (GPDP) [102] and the General Vehicle Routing Problem (GVRP) [41] transportation requests may include multiple pickup and delivery tasks. These problems have in common that all tasks belonging to one transportation request must be executed by the same vehicle. In the mobile workforce scheduling problem considered in this work, however, tasks belonging to the same process may be executed by different resources.

If mobility is omitted from the problem domain, the resulting problem can be interpreted as resource-scheduling and resource distribution in business process management. The foundations of resource-scheduling research date back to the 1950s while considerable progress was made during the 1990s by Kolisch in [66] and Brucker et al. in [15]. Their research introduces resource scheduling from an operations research point-of-view originating in the abovementioned VRP research. Sprecher and Drexl [111] introduce a solution algorithm for projects with precedence constraints. In the realm of business process management such projects can be compared to processes. Ursu et al. [118] present a distributed solution for workforce allocation based on independent agents. The workforce allocation is achieved by negotiation between agents utilizing a specialized communication protocol. Russell et al. introduce a series of 43 workflow resource patterns in [100]. A discussion of organizational aspects of resource management is given in [136]. Netjes et al. [83] introduce a CPN based model to analyze resource utilization and perform several examinations regarding skill balancing and resource allocation order in banking processes. In-depth Petri net based modeling and analysis of work distribution mechanisms of the workflow management systems Staffware, FileNet, and FLOWer are presented in [91]. Further research by Pesic and van der Aalst focuses on the development of a reference model for work distribution in workflow management systems [90]. They concentrate on the general lifecycle of work items and introduce a CPN based approach to distribute work items to resources at runtime. Though most of the work

depicted above had creative influence on our work none covers the properties of mobile process environments.

Resource allocation in mobile process environments has been in the focus of the following work. An automated resource management system (ARMS) for British Telecom is introduced in [127]. The system is intended to forecast and analyze resource demands. It dispatches the jobs to resources, but it does not handle precedence relations of chained tasks and process durations. Cowling et al. introduce a similar problem in [21]. They consider mobile processes with time window restrictions, skill demands, and precedence constraints applying to tasks. The determination of tasks to be performed is based on static priority values of the tasks with the objective to perform a maximum of highly prioritized tasks. Complex cases consisting of several tasks and implications of process durations are not considered. Our problem, in opposition, considers the process as a whole with related constraints.

#### 4.3.1 The Vehicle Routing Problem

The introduction to the VRP given in this section (4.3.1) is a citation of the introduction originally given by Goel in [40]. For the purpose of this work the names and variables of [40] were adapted to match the nomenclature introduced in Chapter 2.

The VRP concerns the distribution of goods and products between a depot and customers. Real-life applications of the VRP are found in the delivery or collection of shipments. The VRP was first presented by Dantzig and Ramser in [23], who describe the problem of delivering gasoline from a bulk terminal to service stations. Other delivery problems arise in various industries, e.g., food and beverages, newspaper, and postal carriers. Collection problems occur e.g. in manufacturing when parts and components have to be transported to the production plant, in the collection of fresh milk, and the collection of garbage. Further information about the VRP, its variants, and applications can be found in the book on VRP edited by Toth and Vigo [115].

Let us consider a set of customers who require the delivery or the collection of a certain shipment. Let  $\mathcal{T}$  denote the set of pickup or delivery tasks at customer sites and let  $\tau_{depot}$  denote the depot where all workers start and end their tour. Let

$$\mathcal{N} := \mathcal{T} \cup \{\tau_{depot}\}$$

denote the set of nodes a worker may travel to and

$$\mathcal{K} := \mathcal{N} \times \mathcal{N} \setminus \{(\tau, \tau) \mid \tau \in \mathcal{T}\}$$

denote the set of edges between any two of the nodes. Each edge  $(n, m) \in \mathcal{K}$  is associated with a nonnegative cost  $\kappa_{nm}$ , which usually represents the

required travel time or travel distance on the edge. Each task  $\tau \in \mathcal{T}$  is associated with a known resource demand or supply  $d_\tau$ . It is assumed that either all tasks  $\tau \in \mathcal{T}$  require a certain amount of goods ( $d_\tau < 0$ ) or request the pickup of a certain amount of goods ( $d_\tau > 0$ ). The capacity of the vehicles is denoted by  $d^{max}$ , and  $d_{\tau_{depot}} \in [0, d^{max}]$  denotes the initial load of the vehicle at the depot. Usually  $d_{\tau_{depot}} = d^{max}$  in delivery problems and  $d_{\tau_{depot}} = 0$  in collection problems.

**Definition 4.5 (VRP tour)** *A sequence  $s = (\tau_1, \dots, \tau_\lambda)$  is a VRP tour if and only if*

- *the tour starts and ends at the depot:  $\tau_1 = \tau_{depot}$  and  $\tau_\lambda = \tau_{depot}$ ,*
- *all tasks to accomplish are well defined:  $\tau_i \in \mathcal{T}$  for all  $1 < i < \lambda$ ,*
- *each task is accomplished at most once: for all  $i, j \in 2, \dots, \lambda - 1$  : if  $\tau_i = \tau_j$  then  $i = j$ .*

**Definition 4.6 (Feasible VRP tour)** *A VRP tour  $s = (\tau_1, \dots, \tau_\lambda)$  is feasible according to capacity constraints if and only if the capacity of the vehicle is large enough to serve all customers of the tour:*

$$0 \leq \sum_{j=1}^{j \leq i} d_{\tau_j} \leq d^{max} \text{ for all } 1 < i < \lambda.$$

The VRP is the problem of finding feasible tours covering all customers such that each customer is visited exactly once and that the cost for operating the tours is minimized. The VRP can be modeled using the two-index binary variables  $x_{nm}$  indicating whether  $m \in \mathcal{N}$  is visited immediately after node  $n \in \mathcal{N}$  by some vehicle ( $x_{nm} = 1$ ) or not ( $x_{nm} = 0$ ). For each node  $n \in \mathcal{N}$  let  $\delta_n$  denote a variable representing the current load at the node.

The Vehicle Routing Problem is minimize

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} \kappa_{nm} \tag{I.1}$$

subject to

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} = \sum_{(m,n) \in \mathcal{K}} x_{mn} \text{ for all } n \in \mathcal{N} \tag{I.2}$$

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} = 1 \text{ for all } n \in \mathcal{T} \tag{I.3}$$



$$\sum_{\substack{(n,m) \in \mathcal{K} \\ n=n_{\text{depot}}}} x_{nm} = |\mathcal{W}| \quad (\text{I.4})$$

$$\delta_{n_{\text{depot}}} = d_{n_{\text{depot}}} \quad (\text{I.5})$$

$$\text{for all } (n, m) \in \mathcal{K} \text{ with } m \in \mathcal{T} : \text{ if } x_{nm} = 1 \text{ then } \delta_m = \delta_n + d_m \quad (\text{I.6})$$

$$0 \leq \delta_n \leq d^{\max} \text{ for all } n \in \mathcal{N} \quad (\text{I.7})$$

$$x_{nm} \in \{0, 1\} \text{ for all } (n, m) \in \mathcal{K} \quad (\text{I.8})$$

The objective function (I.1) represents the accumulated costs for all arcs used in the solution. Equation (I.2) represents the flow conservation constraints which impose that exactly the same number of vehicles reach a task  $\tau \in \mathcal{T}$  as vehicles depart from the same node. Since each task is executed exactly once, this means that exactly one vehicle reaches and leaves this task while all other vehicles neither reach nor leave this task, which is imposed by equation (I.3). Equation (I.4) imposes that all workers leave the depot. Note that  $(\tau_{\text{depot}}, \tau_{\text{depot}}) \in \mathcal{K}$ ; and thus, workers do not have to visit any customer location. Constraints (I.5), (I.6), and (I.7) are the capacity constraints which impose that the accumulated load is within the capacity of the vehicle at each node. Eventually, constraint (I.8) imposes that all values of  $x_{nm}$  are binary.

This two-index formulation is the most basic formulation of the VRP, and many variants of the VRP have been proposed in the literature to consider additional real-life requirements. For this work, the most important variant of the VRP is the VRPTW which introduces the need to serve certain tasks during predefined periods of time.

#### 4.3.2 The VRP with Time Windows

The introduction to the VRPTW given in this section (4.3.2) is a citation of the introduction originally given by Goel in [40]. For the purpose of this work the names and variables of [40] were adapted to match the nomenclature introduced in Chapter 2.

The VRPTW is a generalization of the VRP in which each customer  $\tau \in \mathcal{T}$  is associated with a time interval  $[t_{\tau}^{\min}, t_{\tau}^{\max}]$ , called a *time window*. All customers have to be reached within the specified time window. A vehicle

may have to wait on the hip towards a node such that it arrives within the time window of the node. Let  $t_{\tau_{depot}}$  denote a parameter representing the earliest departure time at the depot. For each customer location  $\tau \in \mathcal{T}$  let  $t_\tau$  denote a variable representing the arrival time at the customer. In order to consider the time required to travel from node  $n \in \mathcal{N}$  to node  $m \in \mathcal{N}$ , the parameter  $d_{nm}$  representing the travel time is specified. The service time needed at a customer location may be included in the travel times of arcs leaving the customer. Throughout this work it is assumed that travel times are positive and obey the *triangle inequality*

$$d_{ij} + d_{jk} \geq d_{ik} \text{ for all } i, j, k \in \mathcal{N}.$$

The triangle inequality expresses that it is never faster to travel between any two nodes by visiting an intermediate node as to directly travel from one node to another. This assumption has little effect on generality, as this property is usually satisfied in real-life applications.

**Definition 4.7 (VRP tour feasible according to time windows)** *A VRP tour  $s = (\tau_1, \dots, \tau_\lambda)$  is feasible according to time windows if and only if arrival times  $t_{\tau_2}, \dots, t_{\tau_{\lambda-1}}$  exist such that*

$$t_{\tau_i} + d_{\tau_i \tau_{i+1}} \leq t_{\tau_{i+1}} \text{ for all } 1 \leq i < \lambda - 1$$

and

$$t_{\tau_i}^{min} \leq t_{\tau_i} \leq t_{\tau_i}^{max} \text{ for all } 1 < i < \lambda.$$

The arrival times of a VRP tour  $s = (\tau_1, \dots, \tau_\lambda)$  can be calculated in  $\mathcal{O}(\lambda)$  time by

$$t_{\tau_{i+1}} := \max(t_{\tau_{i+1}}^{min}, t_{\tau_i} + d_{\tau_i \tau_{i+1}}) \text{ for all } 1 \leq i < \lambda - 1.$$

The Vehicle Routing Problem with Time Windows is minimize

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} \kappa_{nm} \tag{II.1}$$

subject to

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} = \sum_{(m,n) \in \mathcal{K}} x_{mn} \text{ for all } n \in \mathcal{N} \tag{II.2}$$

$$\sum_{(n,m) \in \mathcal{K}} x_{nm} = 1 \text{ for all } n \in \mathcal{T} \quad (\text{II.3})$$

$$\sum_{\substack{(n,m) \in \mathcal{K} \\ n = n_{\text{depot}}}} x_{nm} = |\mathcal{W}| \quad (\text{II.4})$$

$$\delta_{n_{\text{depot}}} = d_{n_{\text{depot}}} \quad (\text{II.5})$$

$$\text{for all } (n, m) \in \mathcal{K} \text{ with } m \in \mathcal{T} : \text{ if } x_{nm} = 1 \text{ then } \delta_m = \delta_n + d_m \quad (\text{II.6})$$

$$0 \leq \delta_n \leq d^{\max} \text{ for all } n \in \mathcal{N} \quad (\text{II.7})$$

$$\text{for all } (n, m) \in \mathcal{K} \text{ with } m \in \mathcal{T} : \text{ if } x_{nm} = 1 \text{ then } t_m \geq t_n + d_{nm} \quad (\text{II.8})$$

$$t_n^{\min} \leq t_n \leq t_n^{\max} \text{ for all } n \in \mathcal{T} \quad (\text{II.9})$$

$$x_{nm} \in \{0, 1\} \text{ for all } (n, m) \in \mathcal{K} \quad (\text{II.10})$$

In this formulation (II.1)–(II.7) and (II.10) are identical to (I.1)–(I.7) and (I.8). Constraints (II.8) and (II.9) are the time window constraints. Constraint (II.8) imposes that each node can only be reached according to the arrival time of the preceding node and the (positive) travel time between the two nodes. Inequality (II.9) imposes that each arrival time is within the time window of the customer.

#### 4.3.3 The Resource-Constrained Project Scheduling Problem

The introduction to the RCPSP given in this section (4.3.3) is mainly a citation of the introductions originally given by Brucker et al. in [15] and by Herroelen et al. in [54]. Also numerous references and literature recommendations can be found there. The names and variables of the original sources were adapted to match the nomenclature of this work.

It is assumed that a project (or case)<sup>1</sup>  $c \in \mathcal{C}$  consists of the tasks

$$\mathcal{T}_c := \{\tau_1, \dots, \tau_n\} \cup \{\tau_0, \tau_{n+1}\}.$$

For the sake of simplicity, in general, a unique dummy beginning task  $\tau_0$  and a unique dummy termination task  $\tau_{n+1}$  are added to the set  $\mathcal{T}_c$ . Frequently, the structure of a case is depicted by a so-called activity-on-node (AON) network, where the nodes and the arcs represent the activities and the precedence relations, respectively.  $G = (\mathcal{T}_c, \mathcal{F}_c)$  denotes the transitively reduced graph of precedence constraints in which  $\mathcal{T}_c$  is the set of vertices (nodes) representing the tasks belonging to the case  $c \in \mathcal{C}$  and

$$\mathcal{F}_c \subset \mathcal{T}_c \times \mathcal{T}_c$$

is the set of edges (arcs) representing the finish-start precedence relationships with zero-time lag.  $\mathcal{F}_c$  can be interpreted as the case's *process flow*. Single precedence constraints between two tasks  $\tau_i, \tau_j \in \mathcal{T}_c$  are denoted by  $(\tau_i, \tau_j)$ . The set of direct predecessors of a task  $\tau \in \mathcal{T}_c$  is defined by  $\text{pred}_c(\tau)$  while  $\text{succ}_c(\tau)$  is the set of direct successors of task  $\tau$ . The tasks are to be performed without preemption. The processing time of a task  $\tau \in \mathcal{T}_c$  is given by  $t_\tau^{\text{work}}$ , its starting time by  $t_\tau^{\text{start}}$  and its finishing time by  $t_\tau^{\text{finish}}$ . There are  $K$  renewable resource types with  $r_{\tau k}$  denoting the constant resource requirement of task  $\tau$  for resource type  $k$  and  $a_k$  denoting the constant availability of resource type  $k$ . The set of tasks in progress in time interval  $]t - 1, t]$  is denoted by

$$\mathcal{T}_t^{\text{active}} = \{\tau \mid t_\tau^{\text{finish}} - t_\tau^{\text{work}} < t \leq t_\tau^{\text{finish}}\}.$$

The Resource-Constrained Project Scheduling Problem is minimize

$$t_{\tau_{n+1}}^{\text{finish}} \tag{III.1}$$

subject to

$$t_{\tau_{n+1}}^{\text{work}} = 0 \tag{III.2}$$

$$t_{\tau_0}^{\text{finish}} = 0 \tag{III.3}$$

$$t_{\tau_j}^{\text{start}} \geq t_{\tau_i}^{\text{finish}} \text{ for all } (\tau_i, \tau_j) \in \mathcal{F}_c \tag{III.4}$$

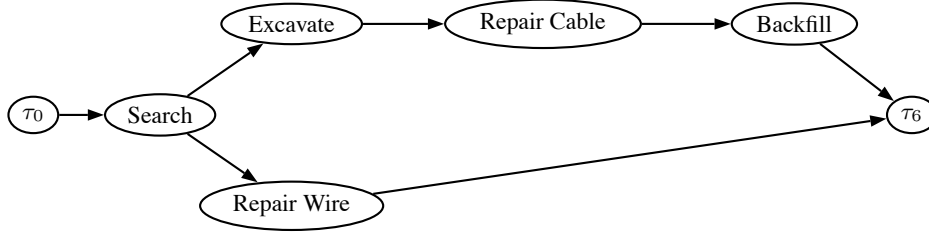
---

<sup>1</sup>In project scheduling theory a set of activities belonging together is usually called a *project* and the activities are usually called *jobs*. To match the terminology of workforce management theory and for the sake of consistency throughout this work, we use the term *case* for project and *task* for job, respectively.

$$\sum_{\tau \in \mathcal{T}_t^{active}} r_{\tau k} \leq a_k \text{ for all } t \leq t_{\tau_{n+1}}^{finish}, k \leq K, t, k \in \mathbb{N}^* \quad (\text{III.5})$$

The objective function (III.1) represents the finishing time of the case's last task to be finished. The duration of the whole case is minimized by minimizing the finishing time of the unique ending dummy task  $\tau_{n+1}$ . Constraint (III.2) assigns a processing time of 0 to the dummy ending task  $\tau_{n+1}$ . Constraint (III.3) assigns a completion time of 0 to the dummy start task  $\tau_0$ . Equation (III.4) represents the precedence constraints indicating that a task  $\tau_j$  must not be started before all of its predecessor tasks  $\tau_i$  are completed. Equation (III.5) represents the resource constraints indicating that for each time period  $]t - 1, t]$  and for each resource type  $k$  the renewable resource amounts required by the tasks in progress cannot exceed the resource availability.

Figure 4.6 gives an example of the RCPSP for the power line damage search and repair process introduced in Section 2.1.3. Note that the time lags of the tasks are omitted in this example to match the formulation of the RCPSP.



$$G = (\{\tau_0, \text{Search}, \text{Excavate}, \text{Repair Cable}, \text{Backfill}, \text{Repair Wire}, \tau_6\}, \\ \{(\tau_0, \text{Search}), \\ (\text{Search}, \text{Excavate}), \\ (\text{Search}, \text{Repair Wire}), \\ (\text{Excavate}, \text{Repair Cable}), \\ (\text{Repair Cable}, \text{Backfill}), \\ (\text{Backfill}, \tau_6), \\ (\text{Repair Wire}, \tau_6)\})$$

Figure 4.6: RCPSP of the power line damage search and repair process

#### 4.3.4 The RCPSP with Preemption

The introduction to the PRCPSPP given in this section (4.3.4) is mainly a citation of the introductions originally given by Brucker et al. in [15] and by Herroelen et al. in [54]. The names and variables of the original sources were adapted to match the nomenclature of this work.

In the RCPSP it is assumed that a task once started must be continuously processed until completion. In practice, however, it may be the case that the processing of a task may be interrupted and resumed at a later time. When resource availability is limited, task preemption may result in a shorter project duration. The introduction of task preemption increases the number of possible solutions and consequently the computational complexity of the RCPSP.

The Preemptive Resource-Constrained Project Scheduling Problem (PRCPSP) allows tasks to be preempted at integer points in time, i.e., the fixed integer duration  $d_\tau = t_\tau^{work}$  of a task  $\tau \in \mathcal{T}_c$  may be split in  $d_\tau$  duration units  $j = 1, 2, \dots, d_\tau$  of length 1. Each duration unit  $j$  of task  $\tau$  is then assigned an integer finish time  $t_{\tau,j}^{finish}$ . The variable  $t_{\tau,0}^{finish}$  denotes the earliest time that a task  $\tau$  can be started. As only finish-start relations with a time lag of zero are allowed,  $t_{\tau,0}^{finish}$  equals the latest finish time of all predecessors of task  $\tau$ . A task  $\tau$  belongs to the set  $\mathcal{T}_t^{active}$  of tasks in progress in period  $[t-1, t]$  if one of its duration units  $j = 1, 2, \dots, d_\tau$  finishes at time  $t$  (i.e.,  $t_{\tau,j}^{finish} = t$ ).

According to [25], the Preemptive Resource-Constrained Project Scheduling Problem is minimize

$$t_{\tau_{n+1},0}^{finish} \quad (IV.1)$$

subject to

$$t_{\tau_{n+1}}^{work} = 0 \quad (IV.2)$$

$$t_{\tau_0,0}^{finish} = 0 \quad (IV.3)$$

$$t_{\tau_j,0}^{finish} \geq t_{\tau_i,d_{\tau_i}}^{finish} \text{ for all } (\tau_i, \tau_j) \in \mathcal{F}_c \quad (IV.4)$$

$$t_{\tau,j-1}^{finish} + 1 \leq t_{\tau,j}^{finish} \text{ for all } \tau \in \mathcal{T}_c, j = 1, 2, \dots, d_\tau \quad (IV.5)$$

$$\sum_{\tau \in \mathcal{T}_t^{active}} r_{\tau k} \leq a_k \text{ for all } t \leq t_{\tau_{n+1},0}^{finish}, k \leq K, t, k \in \mathbb{N}^* \quad (IV.6)$$

The objective function (IV.1) minimizes the makespan by minimizing the earliest start time of the dummy end task which has a processing duration of 0, as declared by constraint (IV.2). Task  $\tau_0$  is assigned an earliest start time

of 0 through constraint (IV.3). Constraint (IV.4) assures that all precedence relations are satisfied: the earliest start time of a task  $\tau_j$  cannot be smaller than the finish time of the last unit of duration of its predecessor  $\tau_i$ . Constraint (IV.5) imposes that the finish time for every unit of duration of a task has to be at least one time unit larger than the finish time for the previous unit of duration. Equation (IV.6) stipulates the resource constraints.

#### 4.3.5 The RCPSP with Multi-Projects

The introduction to the RCMPSP given in the first three paragraphs of this section (4.3.5) is mainly a citation of the introduction originally given by Krüger and Scholl in [69].

The Resource Constrained Multi-Project Scheduling Problem (RCMPSP) as an extension of the RCPSP is considered as the simultaneous scheduling of two or more projects which demand the same scarce resources. Precedence constraints are defined only within projects. Projects are linked by the usage of the same restricted resources of a company. An objective function on company level often has to be considered although objectives of single projects may also be regarded [70]. The company objective as, e.g., maximizing profit is usually aimed at by managing the whole project portfolio or multi-project of the company by a resource manager, whereas project targets are set by single project managers. The latter aim to minimize project delay, project cost, etc.

Multi-project scheduling has been studied not nearly as comprehensively as single-project scheduling. One may distinguish two main research fields in multi-project scheduling—the static and the dynamic project environment [29]. The static environment view assumes a closed project portfolio. All projects of the company are summarized to a super-project (portfolio) and scheduled once. The multi-project is unequivocal, and no rescheduling is necessary. After the last project of a multi-project has been completed, a new multi-project may start.

On the contrary, the dynamic environment view considers an open project portfolio. While scheduled projects are executed, new projects arrive at the system and have to be integrated because the portfolio is changing over time. Rescheduling the system becomes necessary frequently. Research in this field mainly focuses on the static environment. Since this work focuses on mobile *dynamic* environments, a brief overview of prior research of dynamic environments is given here. Dynamic environments are researched by Dumond and Mabert [29]. Their study is based on priority rules for static environments. Due date assignment rules, which are tested by simulation, are added. Dumond [28] as well as Dumond and Dumond [27] extend the former study by introducing different resource availability levels. Bock and Patterson [11]

allow resource preemption in the multi-project. Yang and Sum [134; 135] give attention to dynamic project environments by establishing a dual-level management structure for assigning resources to projects on a higher level and operative project scheduling on a lower level. Ash and Smith-Daniels [5] put emphasis on the learning, forgetting, and relearning cycle in dynamic multi-project environments while Anavi-Isakow and Golany [3] apply queuing theory and adapt the production management concept of CONWIP (CONstant Work In Progress) to the multi-project environment.

For the RCPSP, the dominant objective is to minimize the total duration of the project, i.e., the makespan, while for the RCMPSP, papers minimizing the average or sum of delays are dominant [43]. Further possible objectives deal with the utilization of the workforce, i.e., they focus on the minimization of resources' idle times. Due to the different scheduling objectives, the following notation of the RCMPSP will thus give a minimal set of constraints and present two possible objective functions instead of formulating a dedicated problem. Further problem notations can easily be achieved by extending the set of constraints.

Let  $\mathcal{C}$  be the set of all cases<sup>2</sup> and  $c \in \mathcal{C}$  be a single case. Let

$$n_c := |\mathcal{T}_c| - 2$$

be the number of tasks of case  $c \in \mathcal{C}$  without dummy tasks. Let further

$$\mathcal{T} := \bigcup_{c \in \mathcal{C}} \mathcal{T}_c \text{ and } \mathcal{F} := \bigcup_{c \in \mathcal{C}} \mathcal{F}_c$$

be the sets of all tasks and precedence constraints of all cases  $c \in \mathcal{C}$ .

The Resource Constrained Multi-Project Scheduling Problem is

$$\Phi(\mathcal{T}, \mathcal{F}) \tag{V.1}$$

subject to

$$\text{for all } c \in \mathcal{C}, \tau \in \mathcal{T}_c : t_{\tau_{n_c+1}}^{work} = 0 \tag{V.2}$$

$$\text{for all } c \in \mathcal{C}, \tau \in \mathcal{T}_c : t_{\tau_0}^{finish} = 0 \tag{V.3}$$

$$t_{\tau_j}^{start} \geq t_{\tau_i}^{finish} \text{ for all } (\tau_i, \tau_j) \in \mathcal{F} \tag{V.4}$$

---

<sup>2</sup>Again, we stick to the term *case* instead of project.



$$\sum_{\tau \in \mathcal{T}_t^{active}} r_{\tau k} \leq a_k \text{ for all } t \leq t_{\tau_{n+1}}^{finish}, k \leq K, t, k \in \mathbb{N}^* \quad (\text{V.5})$$

The objective function (V.1) depends on the optimization objectives. Examples are the minimum delay over all cases

$$\text{Min} \sum_{c \in \mathcal{C}} \sum_{\tau \in \mathcal{T}_c} (t_{\tau_j}^{start} - t_{\tau_i}^{finish}) \text{ for all } (\tau_i, \tau_j) \in \mathcal{F} \quad (\text{V.6})$$

and the minimum resource idle time

$$\text{Min} \sum_{t=1}^{t_{\tau_{n+1}}^{finish}} \sum_{k=1}^K \sum_{\tau \in \mathcal{T}_t^{active}} (a_k - r_{\tau k}) \text{ for all } t, k \in \mathbb{N}^*. \quad (\text{V.7})$$

Constraint (V.2) assigns a processing time of 0 to the dummy ending tasks of all cases. Constraint (V.3) assigns a completion time of 0 to the dummy start tasks of all cases. Equation (V.4) represents the precedence constraints indicating that a task  $\tau_j$  must not be started before all of its predecessor tasks  $\tau_i$  are completed. Equation (V.5) represents the resource constraints indicating that for each time period  $]t-1, t]$  and for each resource type  $k$  the renewable resource amounts required by the tasks in progress cannot exceed the resource availability. Note that (V.4) and (V.5) must hold regardless of the number or type of cases to be scheduled.

The problem description of the RCMPSP presented here can be extended to cover task preemption in a straightforward manner.

## 4.4 The Mobile Workforce Scheduling Problem with Multitask-Processes

### 4.4.1 Introduction

In the preceding chapters and sections mobile environments with certain attributes were introduced. To formulate the accordant scheduling problem, at first, a subsumption of these attributes will be given.

**Mobile Case** A mobile case consists of one or more tasks that require workers to travel to a geographic location. The tasks of a mobile case may have finish-start precedence relations assigned. These relations demand one or more tasks to be finished before one or more different tasks can

be started. A mobile case may have a time window associated to it during which all of its tasks must be started and finished. A mobile case is an instance of a mobile process.

**Multi Cases** A number of mobile cases is present at the same time and has to be executed in parallel.

**Mobile Task** A mobile task is an atomic piece of work taking place at a geographic location. It requires certain qualifications of the worker executing it. It may have a time window associated to it during which it must be started and finished. A mobile task is an instance of a mobile activity.

**Mobile Worker** A mobile worker performs mobile tasks of mobile cases. A worker has skills that have to match the qualification requirements of the tasks he performs. A worker is associated with a depot where he begins and ends his working day and he travels between different locations. Based on their respective qualifications and locations, workers may perform not all but just a few tasks of a case, possibly even alternately for two or more cases.

**Process Costs** Executing mobile cases induces costs associated with workers' salaries, travel efforts, violations of time windows, and the like.

**Preemption** During the working day urgent cases may emerge unexpectedly causing the need to assign adequately skilled mobile workers being close-by and, thus, causing numerous mobile cases to be preempted and rescheduled.

Scheduling workers in such environments is a challenging task. We introduce a new generalization of the RCSP, the Mobile Workforce Scheduling Problem with Multitask-Processes (MWSP-MP). This scheduling problem considers costs related to travel efforts, costs related to process execution by differently skilled workers, and process priority constraints.

Consider the power line extension process introduced in Section 2.1.2 (p. 14). Figures 2.2 and 2.3 show the business situation and the associated mobile process. If the whole accordant case is associated with a time window or a maximum duration (as, e.g., for power outages), the cheapest work assignment may require that different workers turn the stations on and off while a third one works at the site of the damage. It is thus necessary to create individual schedules (working plans) for the workers matching the time restrictions of the whole case. For efficiency reasons, additional tasks of different cases are inserted into the schedules. An example for such a setting of two cases is illustrated in Figure 4.7. The respective mobile tasks of both cases are executed by the workers  $w_1$ ,  $w_2$ , and  $w_3$ . The dotted lines illustrate

how the workers are involved in both cases. It is easy to see that the cases become interdependent by the schedules of the workers they have in common. If, e.g., worker  $w_1$  finishes the task “Inspect @L4” later than anticipated by the scheduler, the tasks “Work @L3”, “On @L1”, and “On @L2” may be delayed accordingly.

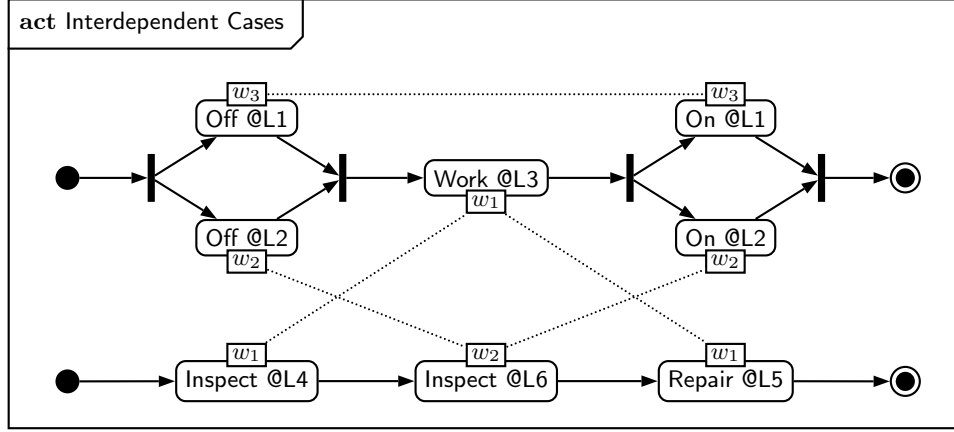


Figure 4.7: Interdependencies of mobile cases

In general, the overlap of concerns of workers’ schedules turns actually independent cases into interdependent cases, because cases as well as traveling/working are subject to time restrictions. For increasing numbers of cases and workers it becomes a challenging task to generate the worker’s schedules. The problem of scheduling mobile workers generalizes both the NP-hard RCPSP and the NP-hard VRP. We present a mathematical formulation of the MWSP-MP by adapting the way the class of RCPSPs is usually formulated [15].

#### 4.4.2 Formulation of the MWSP-MP

In this section, we will introduce the foundations and parameters of the MWSP-MP based on the properties of the processes it is suited for. A preliminary version of the problem formulation has already been published by the author in [42]. We assume that numerous cases are well known in advance and have a long execution horizon (e.g., annual inspections have to be performed in the current year without further timing restrictions given). Such cases usually have a very low priority at the beginning of the year. Additionally, higher prioritized cases show up dynamically and usually have a shorter execution horizon (e.g., same day to one month). An example is the repair of failed equipment. Since cases are subject to cost induced priorities, we want to execute cases with higher priority first. Nonetheless, restrictions

as, e.g., legal regulations may require, that all cases ultimately have to be performed during their respective execution horizon (up to one year).

The planning period is usually too short (one day to one week) to plan all cases present in such organizations. To avoid low priority cases being postponed over and over, we consider to dynamically increase priorities of actually low priority cases gradually whenever the next planning horizon is due. This is subject to the preprocessing of the data before scheduling, not to the scheduling algorithm itself. For any case considered all or none tasks have to be performed.

Based on the nature of the cases in question, the costs of a case are determined by the duration of its execution. For the example in Figure 2.3 (p. 15) this means that the case is more expensive the longer the stations L1 and L2 as well as the according connected consumers are shut down. Further costs arise by the travel times and travel distances of the workforce. Accordingly, we want the workforce to travel as sparse as possible.

Let us denote the set of all cases by  $\mathcal{C}$ . For each process  $c \in \mathcal{C}$  let  $\mathcal{T}_c$  denote the set of tasks belonging to case  $c$  and

$$\mathcal{T} := \bigcup_{c \in \mathcal{C}} \mathcal{T}_c$$

denote the set of all tasks. Each task  $\tau \in \mathcal{T}$  may require some skills (qualifications) for performing the task. These skill requirements are represented by a vector  $q_\tau^{req} := (q_1, \dots, q_k)$ , where  $k$  represents the number of different skills a worker may have.

Let  $\mathcal{F}_c \subset \mathcal{T}_c \times \mathcal{T}_c$  denote the set of precedence constraints associated to case  $c$ . These constraints require that for each pair of tasks  $\tau, \tau' \in \mathcal{T}_c$  with  $(\tau, \tau') \in \mathcal{F}_c$  task  $\tau$  must be completed before the task  $\tau'$  may be started. For each pair  $(\tau, \tau') \in \mathcal{F}_c$  let  $\kappa_{\tau\tau'}^u$  denote the costs arising at each unit of time between the beginning of task  $\tau$  and the completion of task  $\tau'$ . In the power line extension example in Figure 2.3, these costs may represent the costs per unit of time during which the stations L1 and L2 are shut down.

Let us denote the set of all workers by  $\mathcal{W}$ . Each worker  $w \in \mathcal{W}$  has specific skills represented by a vector  $q_w^{avail} = (q_1, \dots, q_k)$ . For each worker  $w \in \mathcal{W}$  let  $\tau_w^{depot}$  denote the worker's depot. Let  $\mathcal{D} := \{\tau_w^{depot} \mid w \in \mathcal{W}\}$  denote the set of all depots. Note that for any two resources  $w, w' \in \mathcal{W}$  we assume that  $\tau_w^{depot} \neq \tau_{w'}^{depot}$ , even if the depot of the two different resources is located at the same geographical position.

Let

$$\mathcal{N} := \mathcal{D} \cup \mathcal{T}$$

denote the set of tasks a worker may perform and

$$\mathcal{K} := \mathcal{N} \times \mathcal{N} \setminus \{(\tau, \tau) \mid \tau \in \mathcal{T}, \tau \notin \mathcal{D}\}$$

denote the set of edges between any two of the tasks. For each worker  $w \in \mathcal{W}$  and each arc  $(\tau, \tau') \in \mathcal{K}$  let  $\kappa_{\tau\tau'}^{travel}$  and  $t_{\tau\tau'}^{travel}$  denote the nonnegative costs and duration for traveling from  $\tau$  to  $\tau'$ . For each worker  $w \in \mathcal{W}$  and each task  $\tau \in \mathcal{T}$  let  $t_{w,\tau}^{work}$  denote the service time worker  $w$  needs for performing task  $\tau$ .

The MWSP-MP is then modeled using the binary variables  $x_{\tau\tau'}^w$  indicating whether worker  $w$  performs task  $\tau$  immediately after task  $\tau'$  ( $x_{\tau\tau'}^w = 1$ ), or not ( $x_{\tau\tau'}^w = 0$ ), and the binary variables  $y_\tau^w$  indicating whether worker  $w$  performs task  $\tau$  ( $y_\tau^w = 1$ ), or not ( $y_\tau^w = 0$ ).

The continuous variables  $t_\tau^{start}$  indicate the work starting time at task  $\tau$ . The continuous variables  $t_\tau^{min}, t_\tau^{max}$  indicate that a time window applies for task  $\tau$ , demanding that the work at task  $\tau$  must not be started before the timestamp  $t_\tau^{min}$  and must not end after the timestamp  $t_\tau^{max}$ .

The resulting bi-objective Mobile Workforce Scheduling Problem with Multitask-Processes is

minimize

$$\begin{aligned} & \sum_{w \in \mathcal{W}} \sum_{(\tau, \tau') \in \mathcal{K}} x_{\tau\tau'}^w \kappa_{\tau\tau'}^{travel} + \\ & \sum_{c \in \mathcal{C}} \sum_{(\tau, \tau') \in \mathcal{T}_c \times \mathcal{T}_c} \sum_{w \in \mathcal{W}} y_{\tau'}^w (t_{\tau'}^{start} + t_{w,\tau'}^{work} - t_\tau^{start}) \kappa_{\tau\tau'}^u \end{aligned} \quad (II.1)$$

maximize

$$\sum_{c \in \mathcal{C}} \pi_c \frac{\sum_{\tau \in \mathcal{T}_c} \sum_{w \in \mathcal{W}} y_\tau^w}{|\mathcal{T}_c|} \quad (II.2)$$

subject to

$$\sum_{(\tau, j) \in \mathcal{K}} x_{\tau j}^w = \sum_{(k, \tau) \in \mathcal{K}} x_{k\tau}^w \text{ for all } w \in \mathcal{W}, \tau \in \mathcal{N} \quad (II.3)$$

$$\sum_{(\tau, \tau') \in \mathcal{K}} x_{\tau\tau'}^w = y_\tau^w \text{ for all } \tau \in \mathcal{N}, w \in \mathcal{W} \quad (II.4)$$

$$y_{\tau_w^{depot}}^w = 1 \text{ for all } w \in \mathcal{W} \quad (II.5)$$

$$\sum_{w \in \mathcal{W}} y_\tau^w \leq 1 \text{ for all } \tau \in \mathcal{N} \quad (II.6)$$

$$\sum_{\tau' \in \mathcal{T}_c} \sum_{w \in \mathcal{W}} y_{\tau'}^w = |\mathcal{T}_c| \sum_{w \in \mathcal{W}} y_\tau^w \text{ for all } c \in \mathcal{C}, \tau \in \mathcal{T}_c \quad (II.7)$$

$$x_{\tau\tau'}^w = 1 \Rightarrow t_{\tau}^{start} + t_{w,\tau}^{work} + t_{\tau\tau'}^{travel} \leq t_{\tau'}^{start} \quad \text{for all } w \in \mathcal{W}, (\tau, \tau') \in \mathcal{K} \mid \tau' \neq \tau_w^{depot} \quad (\text{II.8})$$

$$y_{\tau}^w = 1 \Rightarrow t_{\tau}^{min} \leq t_{\tau}^{start} \leq t_{\tau}^{max} - t_{w,\tau}^{work} \quad \text{for all } w \in \mathcal{W}, \tau \in \mathcal{N} \quad (\text{II.9})$$

$$t_{\tau'}^{start} \geq t_{\tau}^{finish} \quad \text{for all } (\tau, \tau') \in \bigcup_{c \in \mathcal{C}} \mathcal{F}_c \quad (\text{II.10})$$

$$y_{\tau}^w = 1 \Rightarrow q^{req} \leq q^{avail} \quad \text{for all } w \in \mathcal{W}, \tau \in \mathcal{N} \quad (\text{II.11})$$

$$\begin{aligned} x_{\tau\tau'}^w &\in \{0, 1\} \quad \text{for all } w \in \mathcal{W}, (\tau, \tau') \in \mathcal{K} \\ y_{\tau}^w &\in \{0, 1\} \quad \text{for all } w \in \mathcal{W}, \tau \in \mathcal{N} \end{aligned} \quad (\text{II.12})$$

Objective (II.1) is to minimize travel costs plus process execution costs. Note that if worker  $w$  executes task  $\tau'$ ,  $(t_{\tau'}^{start} + t_{w,\tau'}^{work}) - t_{\tau}^{start}$  represents the time between the start of task  $\tau$  and the end of task  $\tau'$  and that  $y_{\tau'}^w = 1$  for at most one worker. Objective (II.2) is to maximize the sum of all priorities associated with processes performed. Equation (II.3) represents the flow conservation constraints forcing that each task  $\tau \in \mathcal{N}$  will be left after being reached by a worker. (II.4) assures that the values of the binary variables  $x_{\tau\tau'}^w$  and  $y_{\tau}^w$  are well defined. (II.5) assures that each worker departs from its depot. (II.6) and (II.7) guarantee that each task is executed at most once and that either all tasks associated to a case are performed or none. Equations (II.8) and (II.9) represent time windows constraints. Equation (II.10) represents the precedence constraints, imposing that a task must not be started before all of its preceding tasks are finished. (II.11) represents skill constraints, imposing that only workers with appropriate qualifications can execute tasks. Note that the operator  $\leq$  is defined to compare vectors element-wise. Finally, equation (II.12) imposes that the values of  $x_{\tau\tau'}^w$  and  $y_{\tau}^w$  are binary.

## 4.5 Solution Methods of Scheduling Problems

The MWSP-MP is—as a generalization of the VRP and the RCPSP—NP-hard. Even for small problem sizes with a small number of cases to be scheduled an optimal solution can thus not be found in reasonable time. Consequently, solving algorithms have to be applied to the MWSP-MP that terminate in reasonable time and produce acceptable results. Such solving algorithms are usually referred to as heuristics.

The introduction to the state of the art of scheduling methods given in this section (4.5) is mostly a citation of the one given by Goel in [40].

A large number of algorithms has been proposed since the Traveling Salesman Problem (TSP) and its successors have been introduced first. Most of these algorithms consider the static variants of these problems, i.e., they assume that the input data is known completely from the beginning and does not change during the allocation and execution process. Recently, the dynamic variants of the scheduling problems gained attention and have been studied more intensely. This section gives an overview of the existing types of algorithms for solving the routing and assignment problems. A full survey of methods for the VRP, the RCPSP, and their variants would be out of the scope of this work. The reader is referred to the book on the VRP by Toth and Vigo [115], as well as the more recent survey on the VRP by Cordeau et al. in [20]. For a survey of the RCPSP the reader is referred to [54] and [67]. The solution methods presented in this section can be classified as assignment methods, construction methods, improvement methods, and meta-heuristics.

#### 4.5.1 Assignment Methods

Assignment methods are methods that assign tasks or transportation requests to vehicles for immediate execution. They are used for highly dynamic problems where the problem data changes very fast and no foresighted planning is likely to perform well. Assignment methods can be used for the VRP and the Full-Truckload Pickup-and-Delivery Problem (FTL PDP), i.e., the special case of the PDP in which all transportation requests are FTL requests. Assignment rules for the FTL PDP that either assign a newly arrived order to an idle vehicle or a vehicle which just becomes available to an open order have been presented by Regan et al. in [95] and more recently by Fleischmann et al. in [32]. Assignment algorithms that simultaneously assign several open orders to idle vehicles are studied by Spivey and Powell in [110] and Fleischmann et al. in [32].

#### 4.5.2 Construction Methods

Construction methods gradually build tours while keeping an eye on the objective function value, but they do not contain an improvement phase per se, see [71]. A comprehensive survey on construction methods for the VRPTW is given by Bräysy and Gendreau in [13]. One of the best-known tour construction methods for the VRP is the savings algorithm by Clarke and Wright in [19]. The algorithm starts by creating a tour for each order and successively merges two tours to form a new tour replacing the former.

The tours are merged with regard to the maximum savings in the objective function. This algorithm naturally applies to problems for which the number of vehicles is not fixed.

Insertion methods are construction methods that successively insert open transportation requests into partially constructed tours. They are well suited for dynamic planning, because they permit to incorporate a new order considering the set of tours which are currently implemented. Insertion methods are very fast and can be used for dynamic vehicle routing problems in which there may not be enough time to employ more sophisticated methods. Furthermore, insertion methods can often be applied to problems incorporating various real-life requirements without losing efficiency. A discussion of efficient insertion methods for vehicle routing problems incorporating complicating constraints can be found in [16]. Early examples of insertion methods have been proposed by Wilson et al. in [132] for the Dynamic Asset Replacement Planning (DARP) and by Solomon in [109] for the VRPTW. Parallel insertion methods for the static VRPTW which simultaneously construct several tours via insertions are proposed by Potvin and Rosseau in [94] and by Antes and Derigs in [4]. Shen et al. presented a computer assistant whose aim is to help dispatchers inserting new transportation requests into existing tours for the PDP in [107]. The computer assistant integrates a learning module based on neural networks which is trained by an expert dispatcher. A graphical user interface allows the dispatchers to make the final insertion decision. Recently Lu and Dessouky patented an insertion method for the Pickup-and-Delivery Problem with Time Windows (PDPTW) which not only considers the classical incremental costs but also the cost of reducing the time window slack so that more opportunities are left for future insertions [76]. Insertion methods for the dynamic PDP are also studied by Yang et al. in [133], Mitrović-Minić in [80] and Fleischmann et al. in [32].

### 4.5.3 Improvement Methods

Many solution techniques for combinatorial optimization problems are based on a simple and general idea. Let  $s$  be a feasible solution of the problem considered and let  $f(s)$  denote the objective function value of  $s$ . For each feasible solution  $s$  the neighborhood of  $s$  is defined by the solutions  $s^*$ , which can be obtained by applying an appropriately defined neighborhood operator to solution  $s$ . So-called local search or neighborhood search methods explore the neighborhood of the current solution  $s$  by searching for a feasible solution  $s^*$  of high quality in the neighborhood of the current solution  $s$ . This solution may be accepted as new current solution; and in this case, the process is iterated by considering  $s^*$  as new current solution. Note that insertion methods can also be interpreted as local search or neighborhood search methods if not all transportation requests must be



served. Concerning maximization (minimization) problems, a new solution  $s^*$  is typically only accepted if  $f(s^*) \geq f(s)$  ( $f(s^*) \leq f(s)$ ). If no solution  $s^*$  with  $f(s^*) > f(s)$  ( $f(s^*) < f(s)$ ) exists in the neighborhood of  $s$ , a local optimum has been reached. A comprehensive work on local search methods is given by Aarts and Lenstra in [1]. A Survey and comparison of local search methods for the VRPTW has been presented by Bräysy and Gendreau in [13].

Improvement methods are local search methods which start with a feasible solution and gradually modify the current solution in order to improve the solution quality. The most simple improvement methods operate on a single tour and optimize the sequence in which the locations are visited. They are often based on methods developed for the TSP, e.g., 2-opt by Lin in [73] and Or-opt by Or in [86]. Others consider several tours simultaneously, e.g., the operators Relocate, Exchange, and Cross originally introduced by Savelsbergh in [101] for the classical VRP. Local optima produced by an improvement method can be very far from the optimal solution, as they only accept solutions that produce an improvement in the objective function value. Thus, the outcome of the improvement heavily depends on the initial solution and the neighborhood definition.

#### 4.5.4 Meta-heuristics

Meta-heuristics are general solution procedures that often embed some of the standard tour construction and improvement methods as well as techniques to escape from local optima of low quality, see [38]. A comprehensive survey on the use of meta-heuristics for the VRPTW is given by Bräysy and Gendreau in [14]. Examples of meta-heuristics are Simulated Annealing, Genetic Algorithms, Ant Systems, Tabu Search, and Iterated Local Search (ILS), see, e.g., [10].

The fundamental idea of Simulated Annealing is to allow moves resulting in solutions of worse quality in order to escape from locally optimal solutions, see [64]. The probability of doing such a move is decreased during search. Although successful for many static problems, it is not clear how to effectively change this probability in dynamic problems, as input data may change during search.

Genetic Algorithms, Ant Systems, and Tabu Search are memory-based methods classified as Adaptive Memory Programming (AMP) methods by Taillard et al. in [114]. Particularly for highly dynamic problems, AMP methods require methods to efficiently update the memory. The memory can only be used effectively if there are only minor changes to the problem data. Examples for AMP methods are the genetic algorithm for the dynamic PDP presented by Pankratz in [87], an ant colony system for the dynamic VRP by

Montemanni et al. in [82], the Tabu Search algorithm for the dynamic VRP by Gendreau et al. in [37], and the Tabu Search algorithms for the dynamic PDP by Gendreau et al. in [36] and Mitrović-Minić in [80].

The essence of ILS is to iteratively build a sequence of solutions generated by an embedded heuristic. It applies the heuristic until it finds a local optimum. Then it perturbs the solution and restarts the heuristic. This generally leads to far better solutions than if one would use repeated random trials of that heuristic, see Lourenço et al. in [75].

Variable Neighborhood Search (VNS) can be interpreted as a specialized ILS based on the idea of systematically changing the neighborhood structure during search, see [81] and [53]. VNS systematically exploits the following observations: a) a local optimum with respect to one neighborhood structure is not necessarily a local optimum for another; b) a global optimum is a local optimum with respect to all possible neighborhood structures; c) for many problems local optima with respect to one or several neighborhoods are relatively close to each other. An example of a VNS algorithm for vehicle routing problems is the algorithm for the multi-depot VRPTW presented by Polacek et al. in [93].

As noted by Ahuja et al. in [2], a critical issue in the design of a neighborhood search approach is the size of the chosen neighborhood. Large neighborhoods increase the quality of the locally optimal solutions, however, locally optimal solutions are difficult to find in very large neighborhoods. In each iteration of the Large Neighborhood Search (LNS) algorithm presented by Shaw in [106] for the VRPTW, customers are first removed from their tours and then re-inserted using a branch-and-bound procedure. In [103] and [99] similar LNS algorithms using fast insertion heuristics for the re-insertion of transportation requests are presented. The use of fast insertion heuristics is more appropriate for dynamic planning as fast response times can be easily achieved. The LNS approach is very well suited for highly constrained vehicle routing problems, see [63], and rich vehicle routing problems in which data may change dynamically, see [41].

## 4.6 Solution Methods for the MWSP-MP

Implementing schedulers for scheduling problems requires to preliminarily define neighborhood operators. Neighborhood operators are used to move from one solution  $s$  in the search space to another solution  $s^*$  in the neighborhood of  $s$ . They are the core of most heuristics and are typically defined in a way that they are easy to calculate and to evaluate [40]. Since the implementation of sophisticated heuristics is out of the scope of this work, two basic neighborhood operators—INSERT and REMOVE—are introduced. Further

operators can be constructed out of these.

#### 4.6.1 Feasibility Criteria for Schedules and Solutions

To define the neighborhood operators, primarily the criteria for the validity (or feasibility) of a solution and the corresponding schedules are defined.

**Definition 4.8 (Schedule feasible according to time windows)** A schedule  $\sigma_w = (\tau_1, \dots, \tau_\lambda)$  of worker  $w \in \mathcal{W}$  is feasible according to time windows if and only if starting times  $t_{\tau_2}^{start}, \dots, t_{\tau_{\lambda-1}}^{start}$  exist such that

$$t_{\tau_i}^{start} + t_{\tau_i}^{work} \leq t_{\tau_{i+1}}^{start} - t_{\tau_{i+1}}^{travel} \text{ for all } 1 \leq i < \lambda - 1$$

and

$$t_{\tau_i}^{min} \leq t_{\tau_i}^{start} \leq t_{\tau_i}^{max} - t_{\tau_i}^{work} \text{ for all } 1 < i < \lambda.$$

**Definition 4.9 (Schedule feasible according to qualifications)** A schedule  $\sigma_w = (\tau_1, \dots, \tau_\lambda)$  of worker  $w \in \mathcal{W}$  is feasible according to qualification constraints if and only if tasks  $\tau_2, \dots, \tau_{\lambda-1}$  exist such that

$$q_{\tau_i}^{req} \leq q_w^{avail} \text{ for all } 1 < i < \lambda.$$

**Definition 4.10 (Feasible schedule)** A schedule is feasible if and only if it is feasible according to

- time windows constraints and
- qualification constraints.

**Definition 4.11 (Solution)** A solution  $s$  of the MWSP-MP is a set of schedules of all workers:

$$s = \bigcup_{w \in \mathcal{W}} \sigma_w.$$

A solution contains exactly one schedule for each worker  $w \in \mathcal{W}$ .

Let  $\mathcal{C}_s \subseteq \mathcal{C}$  be the set of all cases scheduled in the solution  $s$ . Let further be

$$\mathcal{T}_s = \bigcup_{c \in \mathcal{C}_s} \mathcal{T}_c$$

be the set of all tasks scheduled in the solution  $s$ .

**Definition 4.12 (Solution feasible according to precedence)** *A solution  $s$  of the MWSP-MP is feasible according to precedence constraints if and only if tasks  $\tau_i, \tau_j \in \mathcal{T}_s$  exist such that*

$$\text{for all } (\tau_i, \tau_j) \in \mathcal{F} \Rightarrow t_{\tau_j}^{\text{start}} \geq t_{\tau_i}^{\text{finish}}.$$

Note that the starting and finishing times given in Definition 4.12 are time stamps *calculated by the scheduler*. The real time stamps are subject to the operation of the workforce and can thus differ from the values expected. This may cause workers to wait longer or shorter than calculated at scheduling time.

**Definition 4.13 (Feasible solution)** *A solution of the MWSP-MP is feasible if and only if*

- *it is feasible according to precedence constraints and*
- *all of its schedules are feasible.*

Assuming that

$$\text{for all workers } w \in \mathcal{W} : \sigma_w := (\tau_{\text{depot}}, \tau_{\text{depot}}) \text{ is a feasible schedule} \quad (\text{I.1})$$

a feasible initial solution of the MWSP-MP can be obtained by setting the schedule of each worker  $w \in \mathcal{W}$  to  $(\tau_{\text{depot}}, \tau_{\text{depot}})$ . Throughout this work it is assumed that assumption (I.1) is true.

## 4.6.2 Neighborhood Operator INSERT

The generation of solutions requires the insertion of unscheduled cases into existing solutions. The insertion of cases can basically be performed as illustrated in Figure 4.8.

For each case  $c \in \mathcal{C}$  and each task  $\tau \in \mathcal{T}_c$  let us define precedence indices  $\pi_\tau$  in such a way that  $\pi_\tau$  represents the length of the longest path from a task without predecessors to  $\tau$  in the network defined by the set of all tasks  $\mathcal{T}_c$  of  $c$  and the set of all precedence relations  $\mathcal{F}_c$  of  $c$ . Note that all arcs defined by  $\mathcal{F}_c$  are assumed to have length 1. An example of the allocation of precedence indices is illustrated in Figure 4.9 for the activities of the power line extension process introduced in Section 2.1.2.

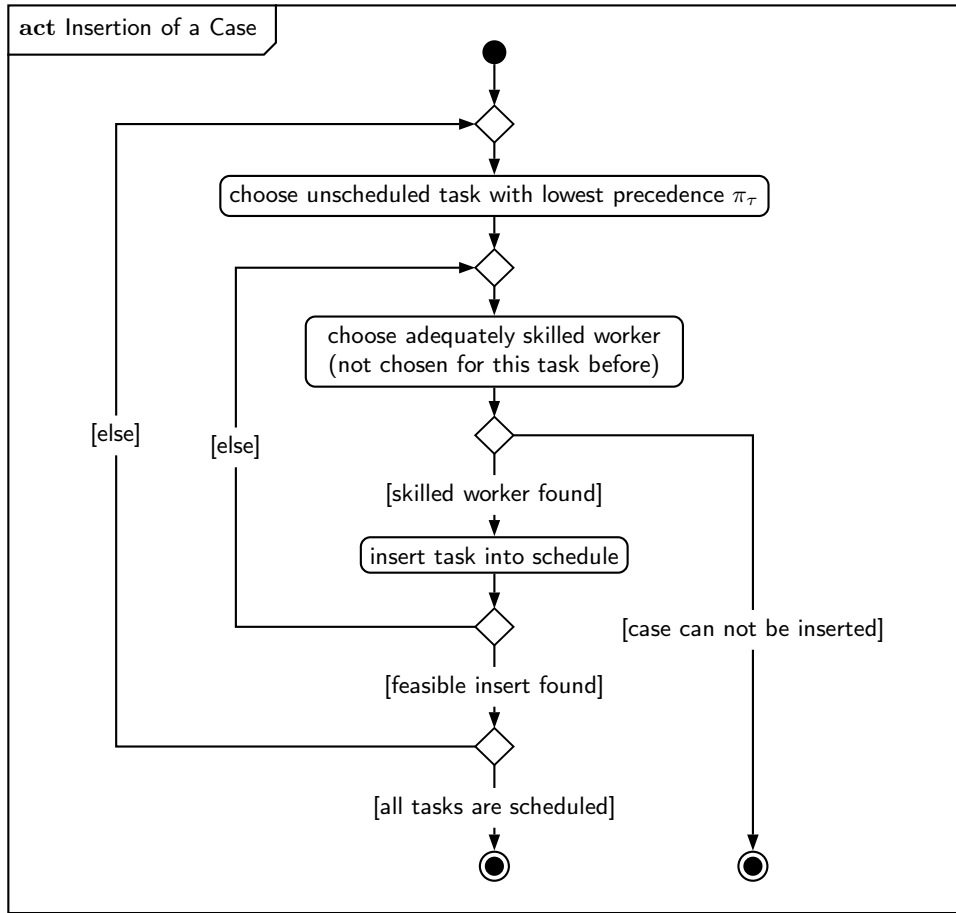


Figure 4.8: Insertion of a case

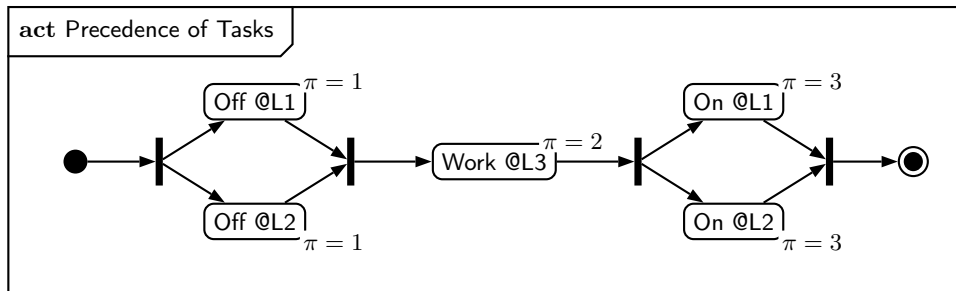


Figure 4.9: Precedence indices of tasks

#### 4.6.2.1 Inserting Tasks into Schedules

Since cases consist of tasks, the step to be defined primarily is the insertion of tasks into schedules of workers. Let  $c \in \mathcal{C}$  be a currently unscheduled case to be inserted into an existing solution  $s$ . Considering that all tasks of  $c$  are

inserted subsequently into schedules  $\sigma$  of  $s$  (see Figure 4.8), let  $\tau^* \in \mathcal{T}_c$  be the task to be inserted next. Let further be  $\mathcal{T}_\sigma$  the set of tasks already inserted into the schedule  $\sigma$ . The insertion of  $\tau^*$  into  $\sigma$  is then defined as follows:

**Definition 4.14 (Insertion of a task)** *An insertion of a task  $\tau^* \in \mathcal{T}_c$  into a schedule  $\sigma$  is a transformation of  $\sigma$  to a schedule  $\sigma^*$  such that  $\sigma$  is a subsequence of  $\sigma^*$  with  $\mathcal{T}_{\sigma^*} = \mathcal{T}_\sigma \cup \{\tau^*\}$ . An insertion is feasible if and only if the resulting schedule is feasible.*

Figure 4.10 illustrates the insertion of an unscheduled task into the schedule  $\sigma$  of a worker.

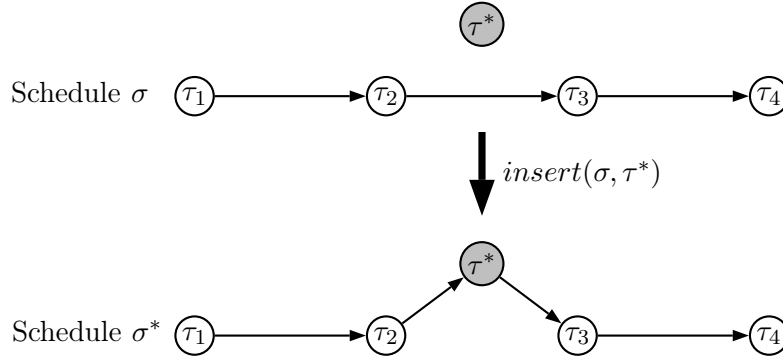


Figure 4.10: Task insertion

To explore the neighborhood defined by the function  $insert(\sigma, \tau^*)$ , a task insertion tree can be constructed for each schedule  $\sigma$  and each unscheduled task  $\tau \in \mathcal{T}_c$ . The root of the task insertion tree corresponds to the starting point of the schedule. At each node the task insertion tree has at most two branches: one for the insertion of the next node belonging to the schedule  $\sigma$  and one for the task  $\tau$ . To cover complete schedules each branch and its subbranches are continued until the final point of the schedule  $\sigma$ . Figure 4.11 shows an example of a task insertion tree for the insertion of a task  $\tau^*$  into a schedule with the initial  $length(\sigma) = 4$ .

The task insertion tree consists of  $length(\sigma) - 1$  leafs indicating that an insertion can basically be performed at each position between any two directly subsequent tasks of  $\sigma$ . It is very likely that not all of the insertions covered by the task insertion tree result in a feasible schedule. For the insertion of cases into solutions a case insertion tree can be constructed accordingly. Figure 4.12 shows an example for the insertion of a case  $c \mid \mathcal{T}_c = \{\tau_a, \tau_b\}$  consisting of two task insertion trees. The connections of the leafs show the pairs of insertion paths leading to schedules with the correct distribution of tasks. The correct distribution of tasks over schedules, however, does not necessarily lead to a feasible solution.

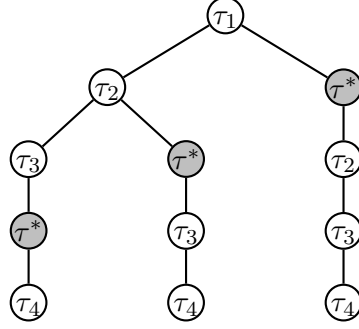


Figure 4.11: Task insertion tree

The insertion's *feasibility according to qualification requirements* can be determined based on the skills of the worker a schedule belongs to. This operation can be performed in constant time for any given worker  $w \in \mathcal{W}$ , and any given task  $\tau^* \in \mathcal{T}$ . Schedules of workers who do not satisfy the qualification requirements of  $\tau^*$  ( $q_w^{avail} < q_{\tau^*}^{req}$ ) can be excluded from further consideration.

The insertion's *feasibility according to time windows* depends on the time window constraints of the already scheduled tasks  $\tau \in \mathcal{T}_\sigma$  of  $\sigma$  and the time window constraints of the task  $\tau^*$  to be scheduled.

Solomon has presented a feasibility criterion concerning time windows for the VRPTW in [109] which was adapted to the GVRP by Goel in [40]. This criterion can be applied to the task insertion of the MWSP-MP as well. Let  $t_{n_1}^{start}, \dots, t_{n_\lambda}^{start}$  be the arrival times of the schedule  $\sigma = (n_1, \dots, n_\lambda)$ . The start times  $t_{m_1}^{start}, \dots, t_{m_{\lambda^*}}^{start}$  of the newly created schedule  $\sigma^* = (m_1, \dots, m_{\lambda^*})$  are calculated by

$$t_{m_j}^{start} = \begin{cases} t_{m_1}^{min}, & \text{for } j = 1, \\ \max(t_{m_{j-1}}^{start} + t_{m_{j-1}}^{work} + t_{m_j}^{travel}, t_{m_j}^{min}), & \text{for all } 1 < j \leq \lambda^*. \end{cases} \quad (\text{II.1})$$

**Definition 4.15 (Slack time of a scheduled task)** *The slack time of a task is the time gap between the arrival at the next task's location and the scheduled start of the next task. This time gap represents the amount of time which the task can be started later than scheduled without affecting the scheduled times and thus the time window constraints of all of its succeeding tasks in the same schedule. The slack time of a task  $\tau_j \in \mathcal{T}_\sigma$  in the schedule  $\sigma$  is the difference between the expected arrival time at the next task  $\tau_{j+1} \in \mathcal{T}_\sigma$  and the expected start time of the next task  $\tau_{j+1}$ :*

$$t_{\tau_j}^{slack} = t_{\tau_{j+1}}^{start} - (t_{\tau_j}^{start} + t_{\tau_j}^{work} + t_{\tau_{j+1}}^{travel}) \text{ for all } j < |\mathcal{T}_\sigma|.$$

**Definition 4.16 (Push forward time of a scheduled task)** *The push forward time represents the time span that a task can be started later than*

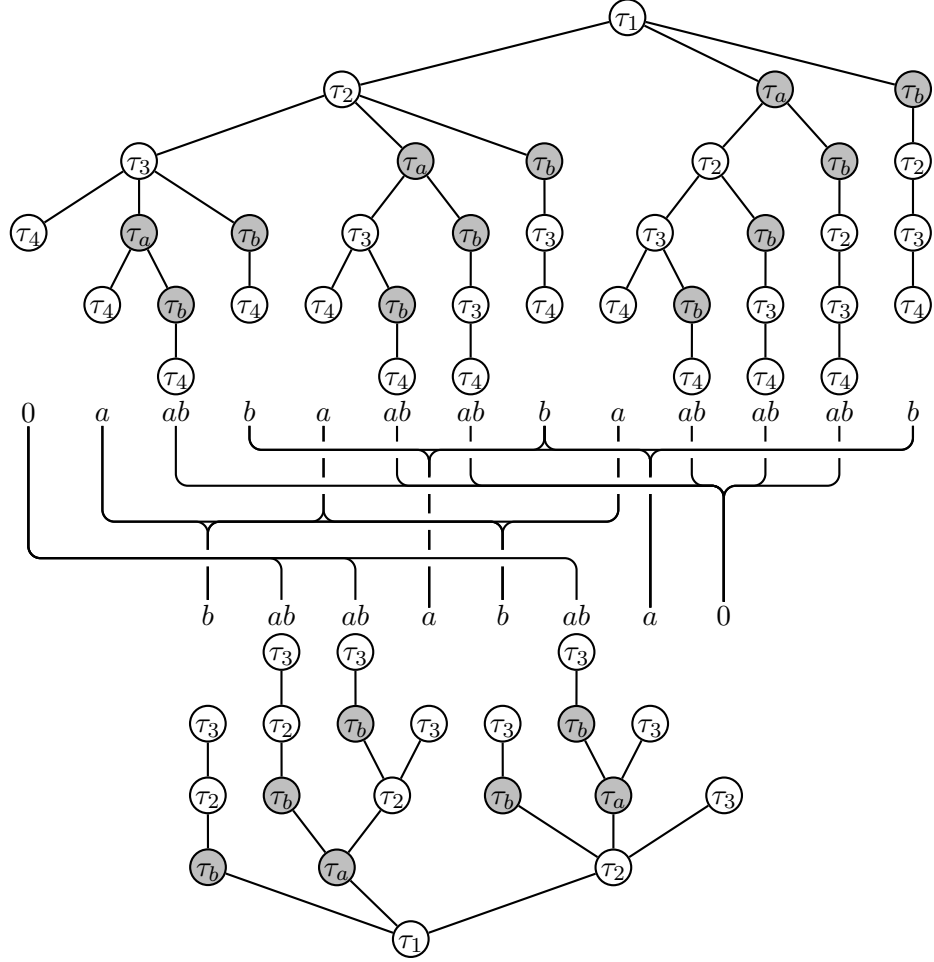


Figure 4.12: Case insertion trees and feasible schedule combinations

scheduled without violating its time window constraints. The push forward time of a task  $\tau_j \in \mathcal{T}_\sigma$  in the schedule  $\sigma$  is defined as:

$$t_{\tau_j}^{push} = \begin{cases} t_{\tau_j}^{max} - (t_{\tau_j}^{start} + t_{\tau_j}^{work}), & \text{for } j = |\mathcal{T}_\sigma|, \\ \min(t_{\tau_j}^{max} - (t_{\tau_j}^{start} + t_{\tau_j}^{work}), t_{\tau_j}^{slack} + t_{\tau_{j+1}}^{push}), & \text{for all } j < |\mathcal{T}_\sigma|. \end{cases}$$

Let  $i \mid 1 < i \leq \lambda$  be the position in the schedule  $\sigma = (n_1, \dots, n_\lambda)$  where the new task  $\tau^*$  is inserted. This operation results in the new schedule  $\sigma^* = (n_1, \dots, n_{i-1}, \tau^*, n_i, \dots, n_\lambda) = (m_1, \dots, m_{\lambda^*})$ . Since exactly one task is inserted per operation  $\lambda^* = \lambda + 1$ .

**Lemma 4.1** *The schedule  $\sigma^* = (m_1, \dots, m_{\lambda^*})$  is feasible according to time window constraints if and only if the starting time of the task directly following the inserted task is less than or equal to the sum of its starting time and*



its push forward time before the insertion:

$$t_{m_{i+1}}^{start} \leq t_{n_i}^{start} + t_{n_i}^{push}.$$

**Proof:**

(I) Let  $i = \lambda$ :

$$\begin{aligned} t_{m_{\lambda^*}}^{start} &\leq t_{n_{\lambda}}^{start} + t_{n_{\lambda}}^{push} \\ &\leq t_{n_{\lambda}}^{start} + (t_{n_{\lambda}}^{max} - (t_{n_{\lambda}}^{start} + t_{n_{\lambda}}^{work})) \\ &\leq t_{n_{\lambda}}^{max} - t_{n_{\lambda}}^{work} \end{aligned}$$

Since  $n_{\lambda} = m_{\lambda^*}$  and  $t^{max}, t^{work}$  are independent of any insertion, this is equivalent to

$$t_{m_{\lambda^*}}^{start} \leq t_{m_{\lambda^*}}^{max} - t_{m_{\lambda^*}}^{work}$$

which must hold for all tasks due to the Definition (4.8).

(II) Let  $1 < i < \lambda$ :

(II.1) Suppose that

$$\begin{aligned} t_{m_{i+1}}^{start} &\leq t_{n_i}^{start} + t_{n_i}^{push} \\ &\leq t_{n_i}^{start} + t_{n_i}^{max} - (t_{n_i}^{start} + t_{n_i}^{work}) \\ &= t_{n_i}^{max} - t_{n_i}^{work}. \end{aligned}$$

Furthermore, due to equation (II.1) we either have  $t_{m_{i+1}}^{start} = t_{m_{i+1}}^{min}$  and feasibility to time windows is trivial, or we have

$$\begin{aligned} t_{m_{i+1}}^{start} &= t_{m_i}^{start} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &\leq t_{n_{i-1}}^{start} + t_{n_{i-1}}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &\leq t_{n_{i-1}}^{start} + t_{n_{i-1}}^{slack} + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_{i-1}}^{start} + t_{n_i}^{start} - (t_{n_{i-1}}^{start} + t_{n_{i-1}}^{work} + t_{n_i}^{travel}) + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_i}^{start} - (t_{n_{i-1}}^{work} + t_{n_i}^{travel}) + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_i}^{start} + t_{n_i}^{push}. \end{aligned}$$

Therefore, the same condition is satisfied for  $i \leftarrow i + 1$ , and it is easy to see that schedule  $\sigma^*$  is feasible according to time window constraints.

(II.2) Suppose that

$$t_{m_{i+1}}^{start} > t_{n_i}^{start} + t_{n_i}^{push}.$$

Then, it is easy to see that the schedule is infeasible according to time windows if  $t_{m_{i+1}}^{start} > t_{m_{i+1}}^{max} - t_{m_{i+1}}^{work}$ . Otherwise, we have

$$t_{n_i}^{max} = t_{m_{i+1}}^{max} \geq t_{m_{i+1}}^{start} + t_{m_{i+1}}^{work} > t_{n_i}^{start} + t_{n_i}^{push}$$

and

$$t_{n_i}^{push} = t_{n_i}^{slack} + t_{n_{i+1}}^{push}.$$

Thus, we have

$$\begin{aligned} t_{m_{i+1}}^{start} &\geq t_{m_i}^{start} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &> t_{n_{i-1}}^{start} + t_{n_{i-1}}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_{i-1}}^{start} + t_{n_{i-1}}^{slack} + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_{i-1}}^{start} + t_{n_i}^{start} - (t_{n_{i-1}}^{start} + t_{n_{i-1}}^{work} + t_{n_i}^{travel}) + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_i}^{start} - (t_{n_{i-1}}^{work} + t_{n_i}^{travel}) + t_{n_i}^{push} + t_{m_i}^{work} + t_{m_{i+1}}^{travel} \\ &= t_{n_i}^{start} + t_{n_i}^{push}. \end{aligned}$$

Therefore, the same condition is satisfied for  $i \leftarrow i + 1$ , and it is easy to see that there exists a  $j \in \{i + 1, \dots, \lambda^*\}$  with  $t_{m_j}^{start} > t_{m_j}^{max}$ . **Q.E.D.**

Since the push times of the original schedule need to be calculated just once, the feasibility according to time windows can be decided in linear time.

#### 4.6.2.2 Precedence Constraints

The feasibility according to precedence constraints depends on the starting times and the finishing times of different tasks of the same case. Since the tasks of one case can be spread over different schedules the feasibility according to precedence constraints can not be decided with respect to a single schedule. An example for the insertion of a case with two tasks  $\tau_1^*$  and  $\tau_2^*$  is depicted in Figure 4.13.

Consider the case insertion scheme depicted in Figure 4.8. To decide whether an insertion is feasible according to precedence constraints not only the currently inserted task has to be considered. If the tasks of a case  $c \in \mathcal{C}$  are inserted into schedules in the order of their respective precedence indices  $\pi$  from lowest to highest, the feasibility according to precedence constraints for a single task  $\tau \in \mathcal{T}_c$  can be achieved by inserting it with a starting time not lower than the largest finishing time  $t_{\tau}^{minins}$  of all tasks of the same case with a precedence  $\pi_{\tau} - 1$ :

$$t_{\tau}^{start} \geq t_{\tau}^{minins} = \begin{cases} t_{\tau}^{start} & \text{if } \pi_{\tau} = 1 \\ \max(t_{\tau^-}^{finish} \mid \tau^- \in \mathcal{T}_c, \pi_{\tau^-} = \pi_{\tau} - 1), & \text{otherwise.} \end{cases}$$

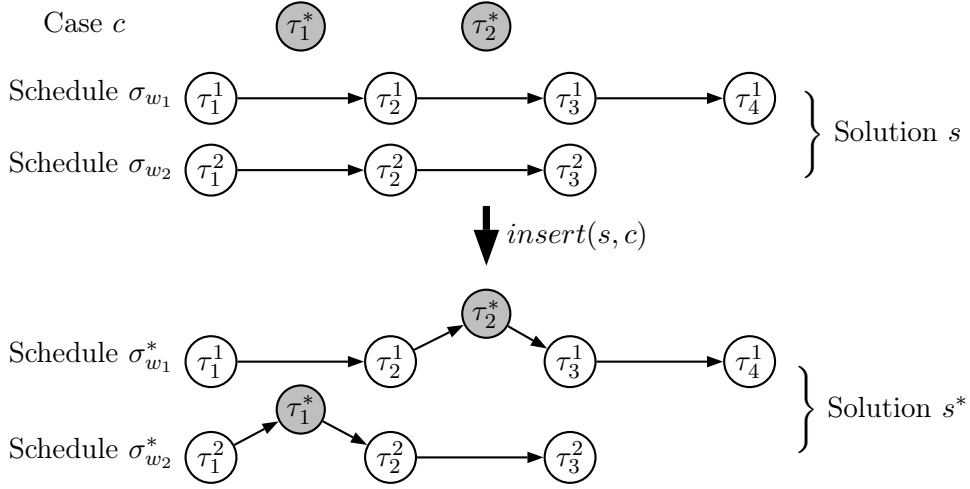


Figure 4.13: Case insertion

Inserting a task  $\tau^*$  into a schedule  $\sigma = (\tau_1, \dots, \tau_\lambda)$  at the position  $i$  leads to a new schedule  $\sigma^* = (\tau_1, \dots, \tau_{i-1}, \tau^*, \tau_{i+1}, \dots, \tau_{\lambda+1})$ . This insertion may cause some or all of the tasks  $\tau_i, \dots, \tau_\lambda \in \mathcal{T}_\sigma$  to be assigned with increased starting times in order to fit  $\tau^*$  into the schedule. The new starting time of a task  $\tau_j \in \mathcal{T}_{\sigma^*}$  is calculated by

$$t_{\tau_j}^{start} = \max(t_{\tau_j}^{start}, t_{\tau_{j-1}}^{start} + t_{\tau_{j-1}}^{work} + t_{\tau_j}^{travel}) \text{ for all } \tau_j \in \mathcal{T}_{\sigma^*}, i < j \leq \lambda$$

to ensure a minimal shifting of succeeding tasks. Additionally, the shift must be feasible according to time window constraints by satisfying Lemma 4.1. For all tasks with adjusted starting times, the feasibility according to precedence constraints has to be evaluated and to be adjusted if necessary. Thus, inserting a case may cascadingly lead to a recalculation of the starting times of many or all tasks already scheduled.

Figure 4.14 outlines the insertion of a task and the Algorithms 1 and 2 depict the cascaded shifting of the starting times with the following function definitions:

- *shift* :  $Schedule \times \mathbb{N} \mapsto Boolean$  Shifts those tasks of the schedule  $\sigma$  with an index  $\geq i$  by adjusting their starting times. Returns *true* if such shifting leads to schedules with fulfilled precedence constraints for all cases, *false* otherwise.
- *adjustCasePrecedence* :  $Case \times \mathbb{N} \mapsto Boolean$  Adjusts the starting times of the tasks of case  $c$  with a precedence index  $\geq \pi$  such that the precedence constraints are fulfilled. Returns *false* if no such adjustment is possible, *true* otherwise.

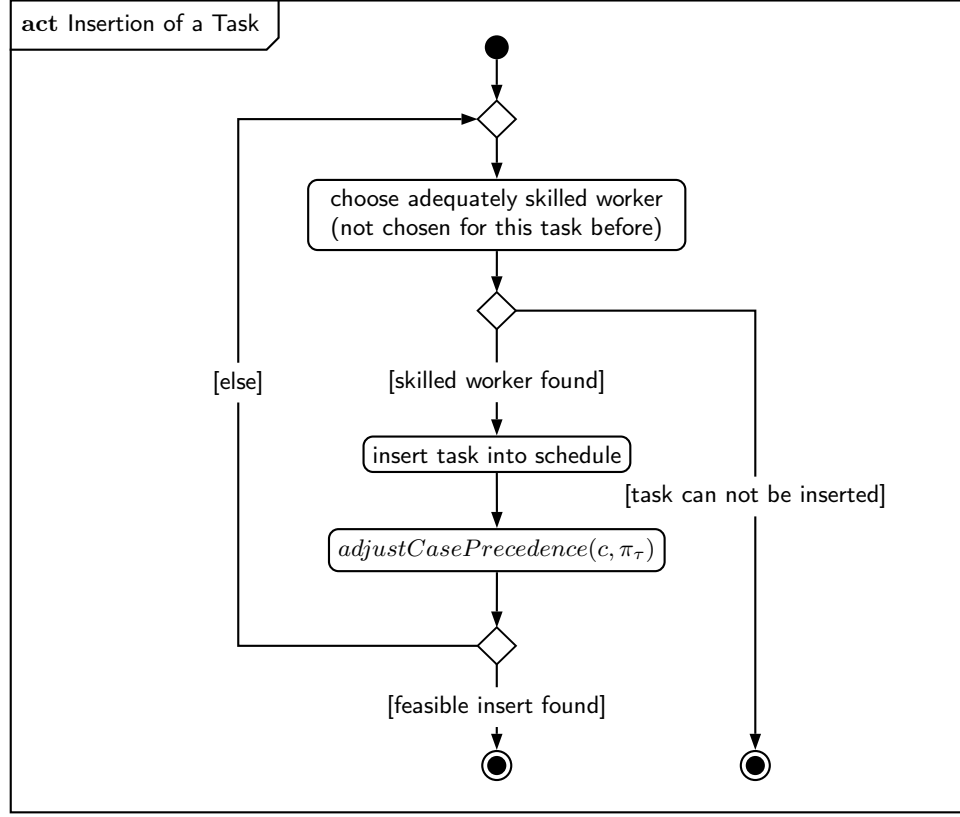


Figure 4.14: Insertion of a task

- $schedule : Task \mapsto Schedule$  Returns the schedule a task is assigned to.
- $index : Task \mapsto \mathbb{N}$  Returns the index (position) of a task in its schedule.
- $case : Task \mapsto Case$  Returns the case a task belongs to.

### 4.6.3 Neighborhood Operator REMOVE

Removing of a case from a set of schedules is the inverse operation of inserting a case.

**Definition 4.17 (Removal of a case)** *The removal of a case  $c \in \mathcal{C}$  from a solution  $s$  is a transformation of  $s$  to a solution  $s^*$  such that  $s^*$  is a subset of  $s$  consisting of schedules being subsequences of the schedules of  $s$  with  $\mathcal{T}_{s^*} = \mathcal{T}_s \setminus \mathcal{T}_c$ . A removal of a case is feasible if and only if the resulting solution is feasible.*

---

**Algorithm 1**  $\text{adjustCasePrecedence}(c, \pi)$ 

---

```

for all  $\tau \in \mathcal{T}_c \mid \pi_\tau = \pi + 1$  do
   $t_\tau^{\text{minins}} \leftarrow \max(t_{\tau^-}^{\text{finish}} \mid \tau^- \in \mathcal{T}_c, \pi_{\tau^-} = \pi)$ 
  if  $t_\tau^{\text{minins}} > t_\tau^{\text{start}}$  then
    if not  $\text{shift}(\text{schedule}(\tau), \text{index}(\tau))$  then
      return false
    end if
  end if
end for
return true

```

---



---

**Algorithm 2**  $\text{shift}(\sigma, i)$ 

---

```

for all  $\tau_j \in \mathcal{T}_\sigma, j > i$  do
   $t_{\tau_j}^{\text{shift}} \leftarrow (t_{\tau_{j-1}}^{\text{start}} + t_{\tau_{j-1}}^{\text{work}} + t_{\tau_j}^{\text{travel}}) - t_{\tau_j}^{\text{start}}$ 
  if  $t_{\tau_j}^{\text{shift}} > 0$  then
    if  $t_{\tau_j}^{\text{shift}} \leq t_{\tau_j}^{\text{push}}$  then
       $t_{\tau_j}^{\text{start}} \leftarrow t_{\tau_j}^{\text{start}} + t_{\tau_j}^{\text{shift}}$ 
      if not  $\text{adjustCasePrecedence}(\text{case}(\tau_j), \pi_{\tau_j})$  then
        return false
      end if
    else
      return false
    end if
  end if
end for
return true

```

---

In coincidence with the insertion of cases into solutions, the removal of cases from solutions is performed task-by-task.

**Definition 4.18 (Removal of a task)** *The removal of a task  $\tau \in \mathcal{T}$  from a schedule  $\sigma$  is a transformation of  $\sigma$  to a schedule  $\sigma^*$  such that  $\sigma^*$  is a subsequence of  $\sigma$  with  $\mathcal{T}_{\sigma^*} = \mathcal{T}_{\sigma} \setminus \{\tau\}$ . A removal of a task is feasible if and only if the resulting schedule is feasible.*

Figures 4.15 and 4.16 illustrate the removal of a task from a schedule  $\sigma$  of a worker and the removal of a case from a solution  $s$ , respectively.

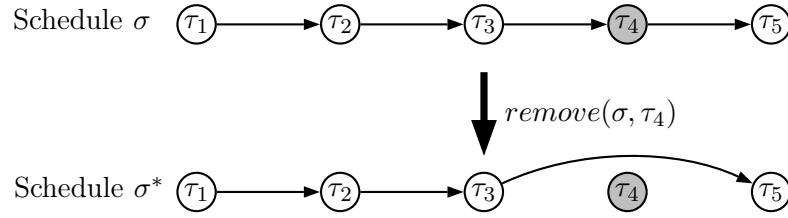


Figure 4.15: Task removal

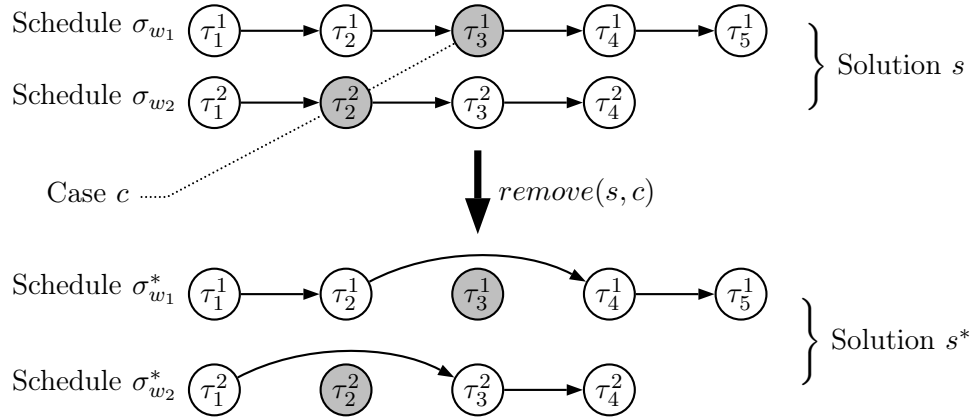


Figure 4.16: Case removal

It is easy to see that after removing a task from a feasible schedule the resulting schedule is feasible. Thus, removing a case from a feasible solution produces a feasible solution. Nonetheless, removing tasks from schedules leads to gaps which may result in increasing worker's waiting times. Additional waiting times can be avoided by either inserting tasks into the gaps or by compacting the schedules. For the former the insertion principles introduced in Section 4.6.2 can be applied while the latter requires additional measures.

Compacting a schedule requires the tasks  $\tau_j$  after the task  $\tau_i$ , that was removed from the schedule  $\sigma^*$ , to be shifted forward in time. Several restrictions

apply to this operation for all tasks  $\tau_j \in \mathcal{T}_{\sigma^*}$ : The minimal time window  $t_{\tau_j}^{min}$ , the earliest possible start  $t_{\tau_{j-1}}^{finish} + t_{\tau_j}^{travel}$ , and the minimal insertion time according to precedence constraints  $t_{\tau_j}^{minins}$ . The resulting new starting time of the task  $\tau_j$  is calculated by

$$t_{\tau_j}^{start} = \max(t_{\tau_j}^{min}, t_{\tau_{j-1}}^{finish} + t_{\tau_j}^{travel}, t_{\tau_j}^{minins}) \text{ for all } \tau_j \in \mathcal{T}_{\sigma^*}, i < j \leq \lambda.$$

After this operation the feasibility according to precedence constraints has to be adjusted as depicted in Algorithm 1.

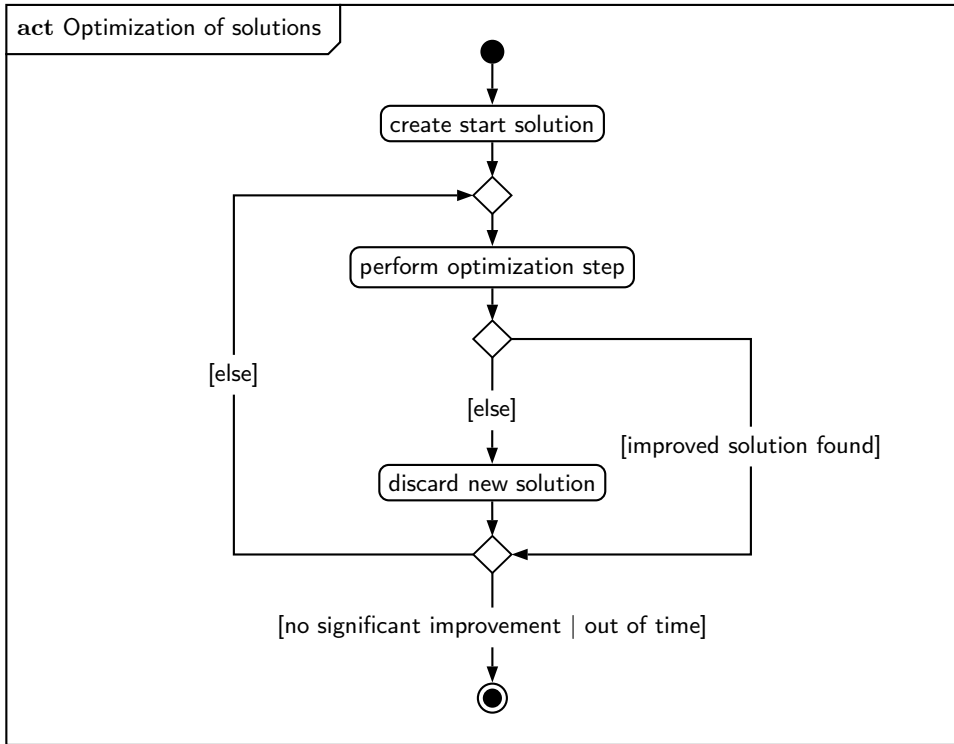


Figure 4.17: Optimization of solutions—general method

#### 4.6.4 Creating Start Solutions

To apply meta-heuristics utilizing the neighborhood operators introduced in Sections 4.6.2 and 4.6.3, it is necessary to generate a start solution for further improvement. The start solutions obtained might then be further improved by meta-heuristic approaches such as LNS [106]. A start solution should satisfy the following requirements: It must contain schedules for all workers, all schedules must cover a full planning period, and all schedules must be feasible. Furthermore, it is desirable to create the start solution in

reasonable time with respect to a good solution quality—e.g., low overall case processing costs. Based on a start solution, improvements can be calculated by meta-heuristics. Figure 4.17 shows the general optimization procedure regardless of the actual method used for optimization.

#### 4.6.4.1 Solution trees

Since a solution consists of several schedules, a number of different possibilities for the insertion of every single task exists. The more schedules and workers are to be considered, the more insertion possibilities exist. Thus, for the incremental insertion of the tasks a tree of solutions evolves.

---

**Algorithm 3** *insert*( $c, s$ )

---

```

 $S := \{s\}$ 
for  $i = 1$  to  $\max\{\pi_\tau | \tau \in \mathcal{T}_c\}$  do
  for all  $\tau \in \mathcal{T}_c$  with  $\pi_\tau = i$  do
     $S^* = \emptyset$ 
    for all  $w \in \mathcal{W}$  with  $q_\tau^{req} \leq q_w^{avail}$  do
      for all  $s \in S$  do
        for  $j = 1$  to  $\lambda_{\sigma_w}^s - 1$  do
           $s^* = \text{insert}(s, \sigma_w, \tau, j)$ 
          if  $s^*$  is feasible then
             $S^* \leftarrow S^* \cup \{s^*\}$ 
          end if
        end for
      end for
    end for
  end for
   $S = S^*$ 
end for

```

---

Creating a start solution begins with an empty solution and inserts cases one after another. To insert cases all possible inserts are performed and the best solution(s) is(are) chosen for further insertions of the following cases. The cases are inserted in the order of their priority from highest to lowest. Algorithm 3 outlines the method<sup>3</sup>.

Inserting cases iteratively into a set of schedules creates two types of solution trees: the first type are single-case trees which contain all possible insertions of the tasks of one single case into the set of schedules. Figure 4.18 shows an example of a single-case tree for a case with two subsequent tasks  $\tau_1$  and  $\tau_2$ . On every level of the tree one task is inserted. Note that the precedence

---

<sup>3</sup>Note that the operator  $\leq$  compares the vectors element-wise.



constraints are satisfied for the solutions framed bold, while they are not necessarily satisfied for the other solutions.

The second type of solution trees are multi-case trees containing tasks of more than one case. During the scheduling of cases and tasks, this type of solution tree is typically created. Figure 4.19 shows a multi-case tree for a case  $c_b \mid \mathcal{T}_{c_b} = \{\tau_{b1}, \tau_{b2}\}$  with the *subsequent* tasks  $\tau_{b1}$  and  $\tau_{b2}$ , a solution set  $s = \{\sigma_1, \sigma_2\}$  of two schedules and one already scheduled case  $c_a \mid \mathcal{T}_{c_a} = \{\tau_{a1}, \tau_{a2}\}$  with the *parallel* tasks  $\tau_{a1}$  and  $\tau_{a2}$ .

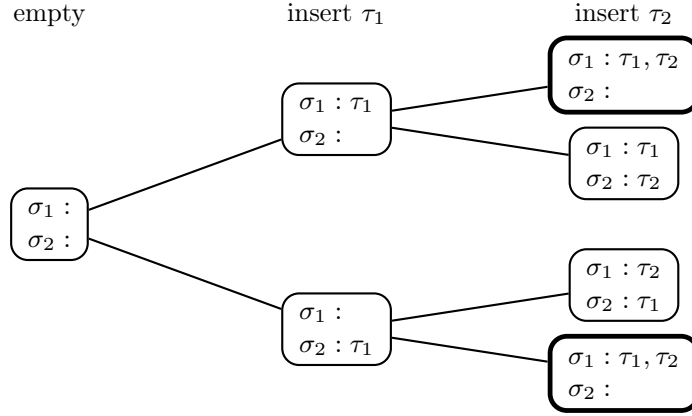


Figure 4.18: Single-case tree example

As for the single-case tree precedence constraints are considered and satisfied locally for every schedule in Figure 4.19. Furthermore, precedence constraints across different schedules are satisfied for the solutions framed bold, while again they are not necessarily satisfied for the other solutions. The latter solutions need further care by the scheduler—e.g., the consideration of travel times or skills.

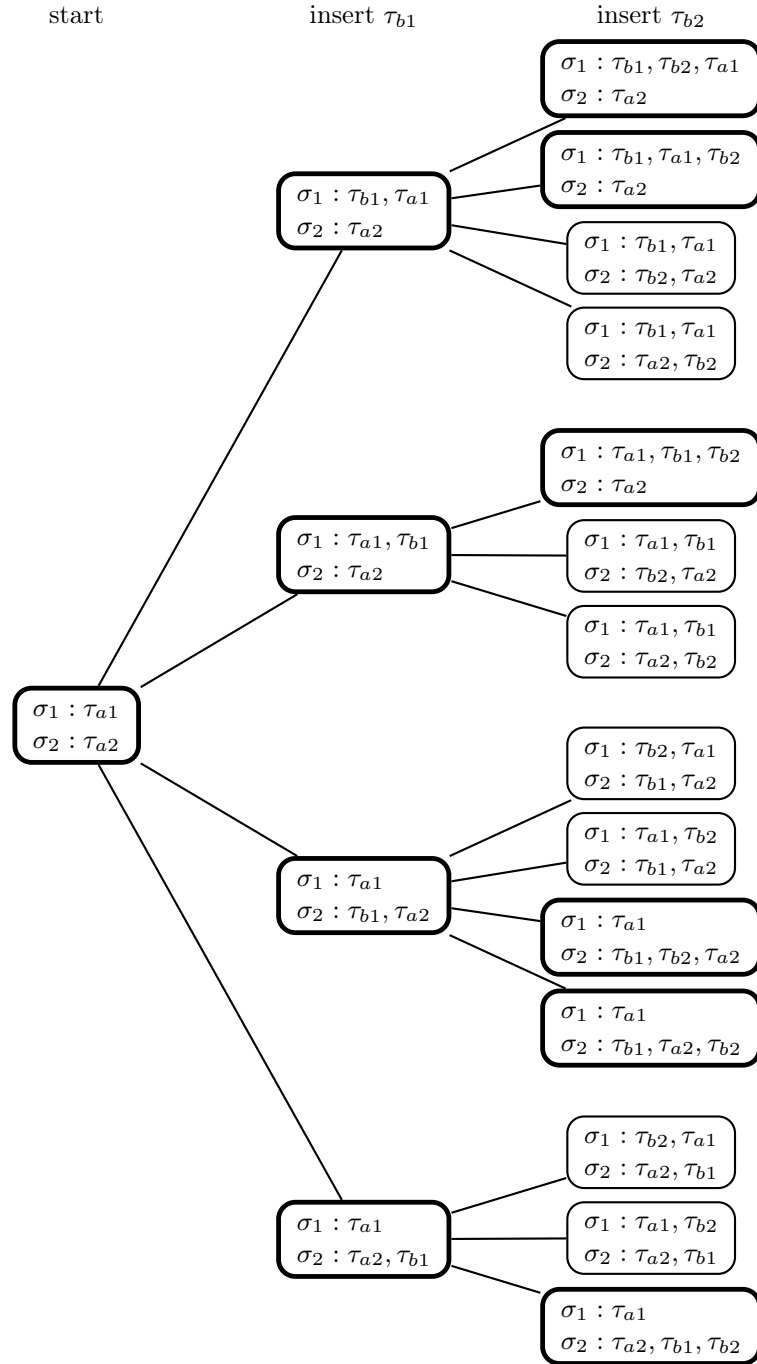


Figure 4.19: Multi-case tree example

#### 4.6.4.2 Reducing the solution tree size

Inserting tasks into schedules subsequently leads to rapidly growing tree sizes—as it is typical for the TSP and its generalizations. To keep the size of the tree handleable, it is necessary to dismiss solutions which have no chance to lead to good solutions. The tree size can be reduced by applying feasibility checks and by the removing branches that appear to be far from optimal in terms of the target cost function. After inserting a task it is thus necessary to remove leafs from the solution tree such that only a certain number of branches remains in the tree. The number of branches to be kept depends on the computational power and the time available for the scheduling. It may vary for different organizations with different mobile environments. Furthermore, it is possible to allow exceptions in the number of branches to be kept due to the quality of the solutions: If a number of branches of the current step has very similar target costs, it may be sensible to keep a larger number of branches, while it may be unnecessary to keep solutions that differ from the better ones by a large cost delta. For the size-aware creation of the solution trees depth-search-first and breadth-search first approaches are considered.

**Depth First Search** The Depth First Search (DFS) algorithm traverses the tree from the root directly to the leaves. After finishing all sub-trees of one level the algorithm tracks back one level.

---

**Algorithm 4**  $dfs(node, task)$ 


---

GLOBAL  $cost \leftarrow \text{MAXNUMBER}$

GLOBAL  $n$

**Function:**  $dfs(node, task)$

create all possible inserts of  $task$  as children of  $node$

calculate cumulated cost for each child

$ntask \leftarrow succ(task)$

**if**  $ntask$  **then**

keep  $n$  cheapest children, skip others

**for all** kept children **do**

$dfs(child, ntask)$

**end for**

**else if** cost of cheapest child  $< cost$  **then**

$cost \leftarrow$  cost of cheapest child

**end if**

---

Algorithm 4 depicts the DFS method in pseudo-code. We assume that the schedules are accessible to the function  $dfs$ . The calculation is started with an empty root node as parameter  $node$  and the first task  $\tau_1$  of the case as parameter  $task$ . The set of the best solutions is determined on every level of

each sub-tree.

Figure 4.20 illustrates the construction of a solution tree by the Depth First Search algorithm. In this example in every step two branches are chosen for further calculation while the rest is skipped ( $n = 2$ ). The circled numbers represent the total cost of the solution with the tasks scheduled at that point. The example was chosen such that the cheapest solution is found in a branch that has not the lowest total cost on every level (cost 3 vs. 2 on level 1).

**Breadth First Search** The Breadth First Search (BFS) algorithm traverses the tree level-wise. It inserts the current task into all branches of the same level before the next task is chosen for insertion.

---

**Algorithm 5**  $bfs(queue, task)$

---

GLOBAL  $cost \leftarrow \text{MAXNUMBER}$

GLOBAL  $n$

**Function:**  $bfs(queue, task)$

$nqueue \leftarrow \emptyset$

**for all** nodes in  $queue$  as  $node$  **do**

    find all possible inserts of  $task$  as children of  $node$

    append children to  $nqueue$

**end for**

calculate cumulated cost for each node in  $nqueue$

$ntask \leftarrow succ(task)$

**if**  $ntask$  **then**

    keep  $n$  cheapest nodes in  $nqueue$ , remove others

$bfs(nqueue, ntask)$

**else if** cost of cheapest node in  $nqueue < cost$  **then**

$cost \leftarrow \text{cost of cheapest node in } nqueue < cost$

**end if**

---

Algorithm 5 depicts the BFS method in pseudo-code. We assume that the schedules are accessible to the function  $bfs$ . The calculation is started with an empty root node as the only node in the parameter  $queue$  and the first task  $\tau_1$  of the case as parameter  $task$ . The set of the best solutions is determined on every level of the whole tree.

Figure 4.21 illustrates the construction of a solution tree by the Breadth First Search algorithm. In this example, on every level two nodes are chosen for further calculation while the rest is skipped ( $n = 2$ ). The circled numbers represent the total cost of the solution with the tasks scheduled at that point.

Note that the example depicted in Figures 4.20 and 4.21 is chosen such that the algorithms find different solutions as best solution with the better one found by Depth First Search. Since the Breadth First Search calculates the

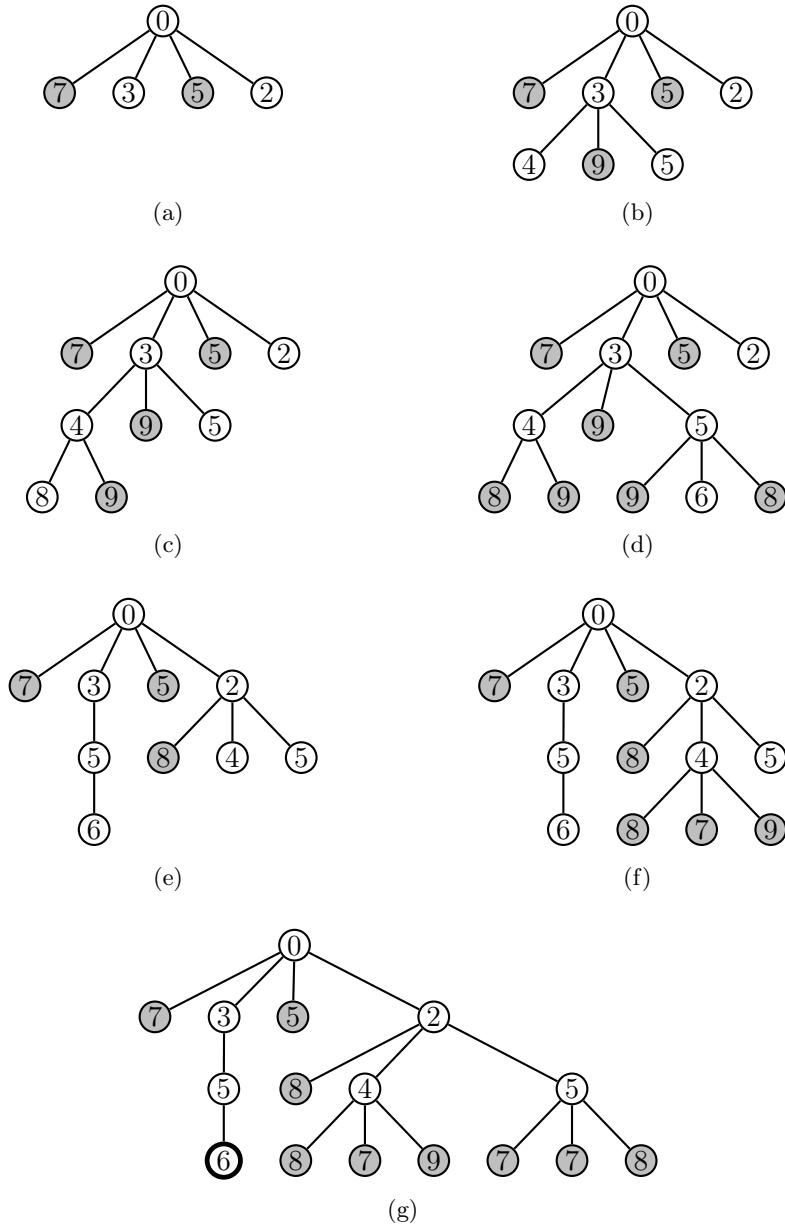


Figure 4.20: Depth First Search example

set of the best solutions less often than the Depth First Search, it tends to skip more solutions at a time. It might be sensible to select the number  $n$  of solutions to be kept larger for Breadth First Search than for Depth First Search.

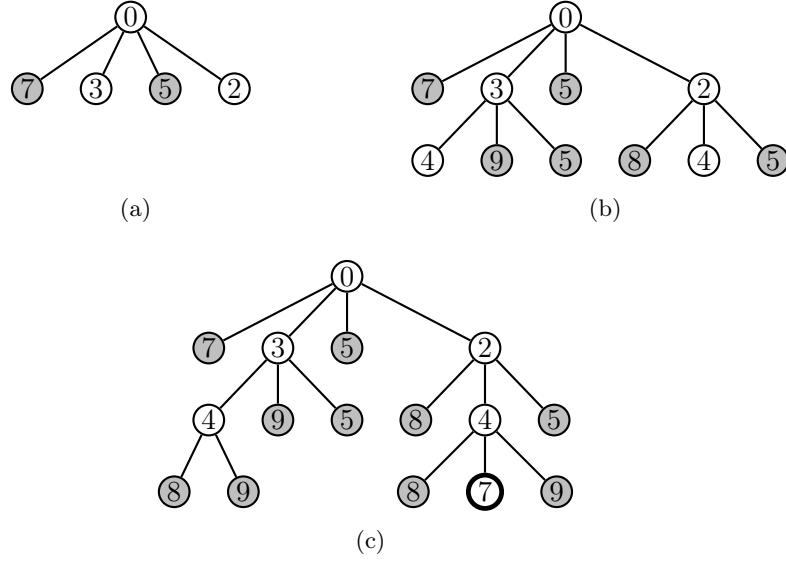


Figure 4.21: Breadth First Search example

**Random tree search** Random tree search is a variant of both search methods introduced above. Instead of always choosing the  $n$  best solutions of the current level, it randomly chooses the solutions to be kept. The degree of randomness may be variegated. For instance, the algorithm could choose the  $n/2$  best solutions of the solution set currently under consideration and additional  $n/2$  random solutions of the same solution set.

## 4.7 Chapter Summary

In this chapter the Mobile Workforce Scheduling Problem with Multitask-Processes and its theoretical foundations were introduced. It was shown that the MWSP-MP is a generalization of the VRP and the RCPSP. Three tree-search methods to generate start solutions of the MWSP-MP were developed based on the neighborhood operators for the insertion and the removal of tasks as well as cases. The methods can be utilized in a mobile environment to create the schedules for mobile workers.

## Chapter 5

# Validation

The aims of this chapter are (i) to validate and compare the algorithms for finding start solutions of the MWSP-MP as introduced in Section 4.6 and (ii) to validate, if the simulation method as introduced in Chapter 3 is a correct representation of mobile work and if it is capable of predicting the outcomes of changes applied to mobile environments. The algorithms for start solutions of the MWSP-MP are compared by scheduling software developed in Java. The CPN models are validated by running business process simulations in CPN Tools. A Java scheduling software is utilized via a TCP interface. All validation is performed on the same subject of study, namely a German gas and power supply, introduced in the first section. In the second section an experimental setup to compare workforce scheduling methods is introduced, and the results of the experiments are presented and discussed. In the third section different evaluation criteria for mobile environments are applied to the real world data and accordant simulation results are presented and discussed. Furthermore, a tool for the setup of simulation experiments is presented.

### 5.1 Subject of Study

#### 5.1.1 ENSO—A German Gas and Power Supply

ENSO is a gas and power supply covering an area of about 7,000 km<sup>2</sup> and serving about 500,000 customers in the East of Germany (see the grayed area in Figure 5.1). Even though several cities are situated in this area, ENSO delivers power and gas only to the rural part of it. The largest town powered by them has about 35,000 inhabitants. In several consulting projects during the years 2004 until 2010 the author analyzed ENSO's business processes in network maintenance.

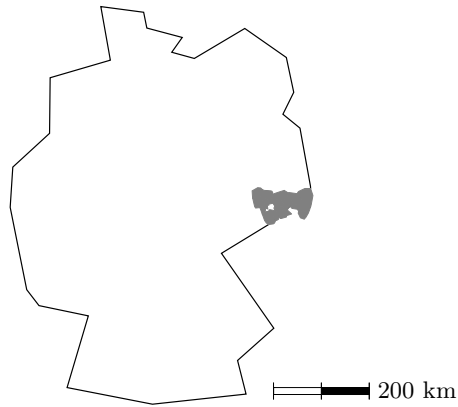


Figure 5.1: ENSO in Germany

To validate the domain model and the procedures that were introduced in the preceding chapters, a subset of ENSO's low voltage (400 V) and medium voltage (10–20 kV) equipment was considered. Table 5.1 gives an overview of the network.

Parameter	Value
Area	approx. 7,000 km <sup>2</sup>
East-West spread	approx. 130 km
North-South spread	approx. 75 km
No. of regions	19
Workforce	134
Length of transmission and supply lines	approx. 10,000 km
No. of equipment sites (substations, branches, distribution stations)	approx. 9,000

Table 5.1: Basic characteristics of ENSO's low and medium voltage network

In order to reduce the computational effort, a subset of the regions was selected for the validation. These regions are marked in Figure 5.2 and are referred to as “East”, “South”, and “West”. Note that the dotted area is not covered by ENSO's services.

Figure 5.3 gives a detailed view of the three regions, the depot, and the equipment sites (assets). The regions cover an area of approximately 920 km<sup>2</sup>. The crossed circle in the center marks the *common* depot of all three regions. The regions share the depot, which is due to historic reasons: firstly, the depot is situated in the outskirts of the largest town of the area; and secondly, the regions used to be one large region formerly. Table 5.2 gives an overview of the regions' characteristics.



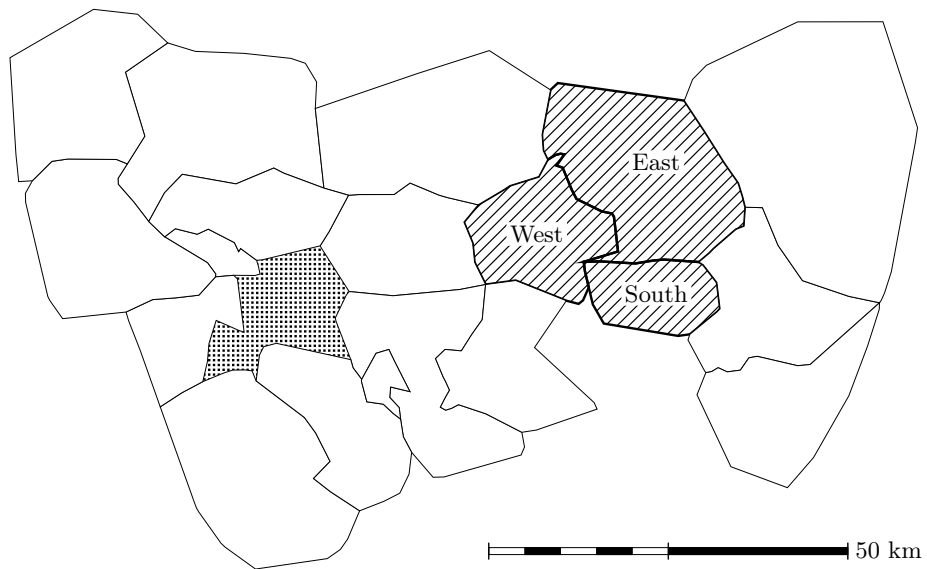


Figure 5.2: Regions of ENSO and selection



Figure 5.3: Selected regions in detail with assets and depot

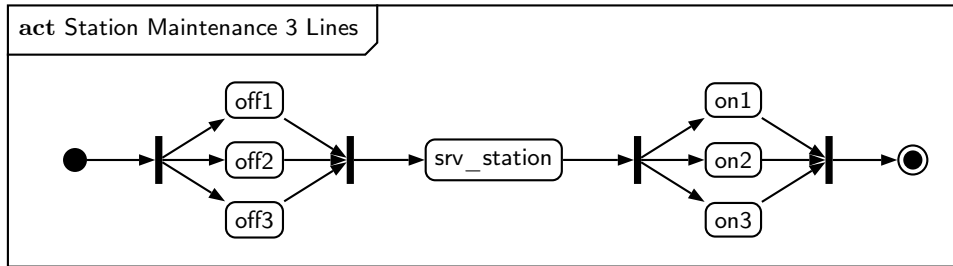
Parameter	East	South	West	Total
Area (approx. km <sup>2</sup> )	490	170	260	920
Foremen	2	2	2	6
Workers	5	5	6	16
No. of medium voltage line segments	428	333	382	1143
No. of medium voltage assets	404	310	370	1084
Assets per km <sup>2</sup>	1.70	3.78	2.89	2.42

Table 5.2: Characteristics of the selected regions

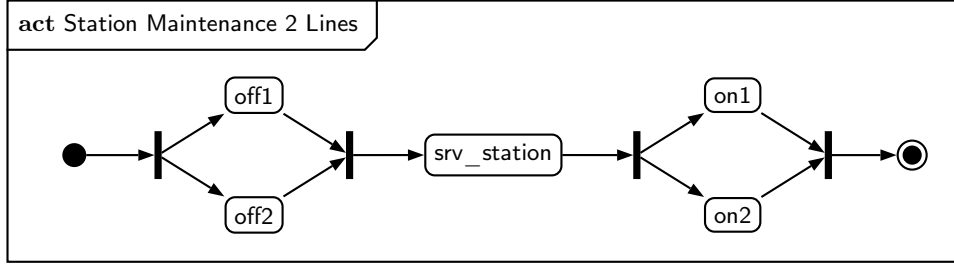
### 5.1.2 Processes and Cases

During a consulting project for ENSO, the author of this work issued a study [97] examining the medium voltage network maintenance processes. The project brought up the structure of the respective processes and their execution incidence for one of ENSO's regions. Since the types of technical equipment and the processes are identical for all regions, a set of typical processes for the selected regions could be derived from the data of the study. These processes were considered as templates to create the processes and cases used to validate the model as described in the following list:

**Station Maintenance 3 Lines** This process is performed to maintain a power substation. During the maintenance work the station and its connecting lines have to be switched off, which is performed at all of its neighboring stations. For this process it is assumed that the station has three neighbors. The cases of this process are performed at four different locations (off1/on1, off2/on2, off3/on3, srv\_station).



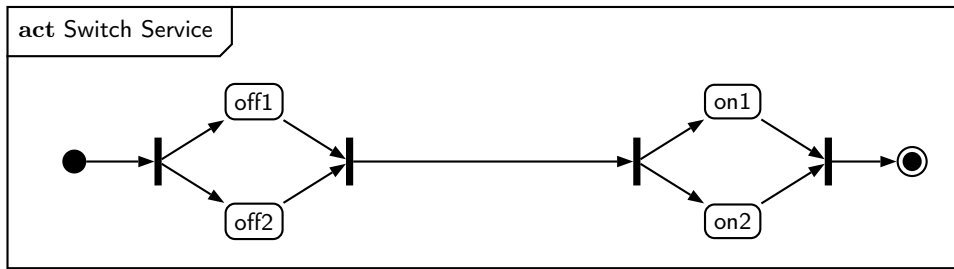
**Station Maintenance 2 Lines** This process is almost identical to the process “Station Maintenance 3 Lines”. The only difference is that the power substation in question has only two neighboring stations. The cases of this process are performed at three different locations (off1/on1, off2/on2, srv\_station).



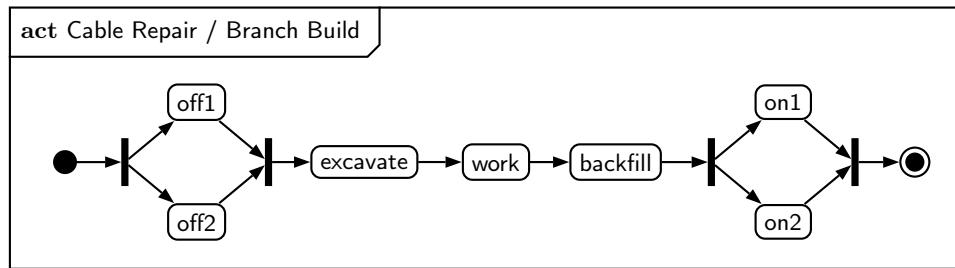
**Switch Service** Besides their actual maintenance work, the network service department acts as a service provider for other departments of the company, namely, for the construction and the power transmission departments. These departments occasionally demand different network setups for their tasks, which is obtained by switching lines in the field. Usually, the switchings are reversed after the other departments have finished their work. To enforce the delay between the switchings—necessary for the work of the other departments—the switching operations are associated with time windows. Three different sets of time windows were evenly assigned to the Switch Service cases as follows (times in hours):

- $[t_{on1/2}^{min}, t_{on1/2}^{max}] = [0, 1]; [t_{off1/2}^{min}, t_{off1/2}^{max}] = [4, 100]$
- $[t_{on1/2}^{min}, t_{on1/2}^{max}] = [0, 2]; [t_{off1/2}^{min}, t_{off1/2}^{max}] = [5, 7]$
- $[t_{on1/2}^{min}, t_{on1/2}^{max}] = [2, 4]; [t_{off1/2}^{min}, t_{off1/2}^{max}] = [6, 100]$

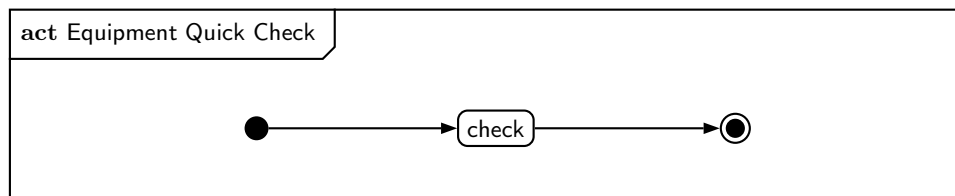
A Switch Service case is performed at two different locations (off1/on1, off2/on2).



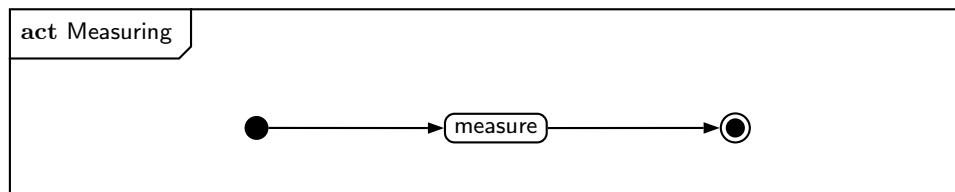
**Cable Repair** This process models the repair of a buried power cable. Excavation work at constructions sites of third parties may lead to cable damages requiring repair. Additionally, building cable branches for network extension (for an example see Figure 2.2 (p. 15)) or for home builders requires the same process. The cases of this process are performed at three different locations (off1/on1, off2/on2, excavate/work/backfill).



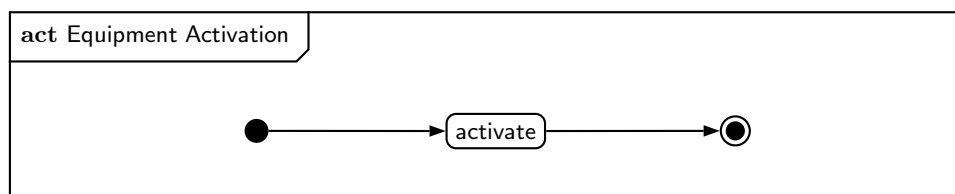
**Equipment Quick Check** This process is performed to check the state of a technical equipment. As the processes introduced further down, it is performed at one single site and consists of just one single task. Thus, the respective cases are simple cases.



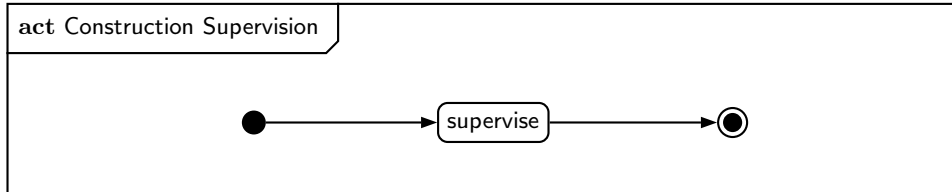
**Measuring** This process is performed to measure operating factors at the site of a technical equipment.



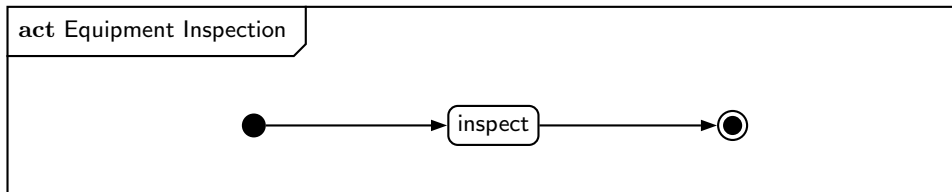
**Equipment Activation** This process is performed after a new technical equipment was built. The process includes testing the asset, connecting it to the network, and turning it on.



**Construction Supervision** The construction of new technical equipment is often performed by contractors. During this process their work is being supervised.



**Equipment Inspection** This process is similar to “Equipment Quick Check” but is defined more formally; it thus requires more time to be performed.



**Customer Site Service** This process covers all types of tasks that can be performed at the site of a customer. This may include the maintenance and repair of equipment, the activation or deactivation of a customer’s power hook-up, or the information of customers about planned power outages.

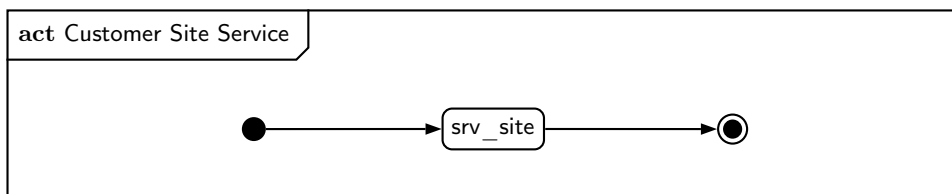


Table 5.3 gives an overview of the qualifications required for each task, the respective precedences of the tasks with respect to different processes, and the typical duration for the execution of a task. Four different qualifications were defined: Maintenance, Electrician, Earthwork, and Foreman. The ordinary workers have the former three qualifications, while the foremen have the latter three qualifications.

The initial set of cases is based on the number of assets of each region (compare Table 5.2). The number of cases of each process was calculated based on the results of the consulting project [97] such that the case-to-asset ratio is almost identical throughout the regions. The total number of

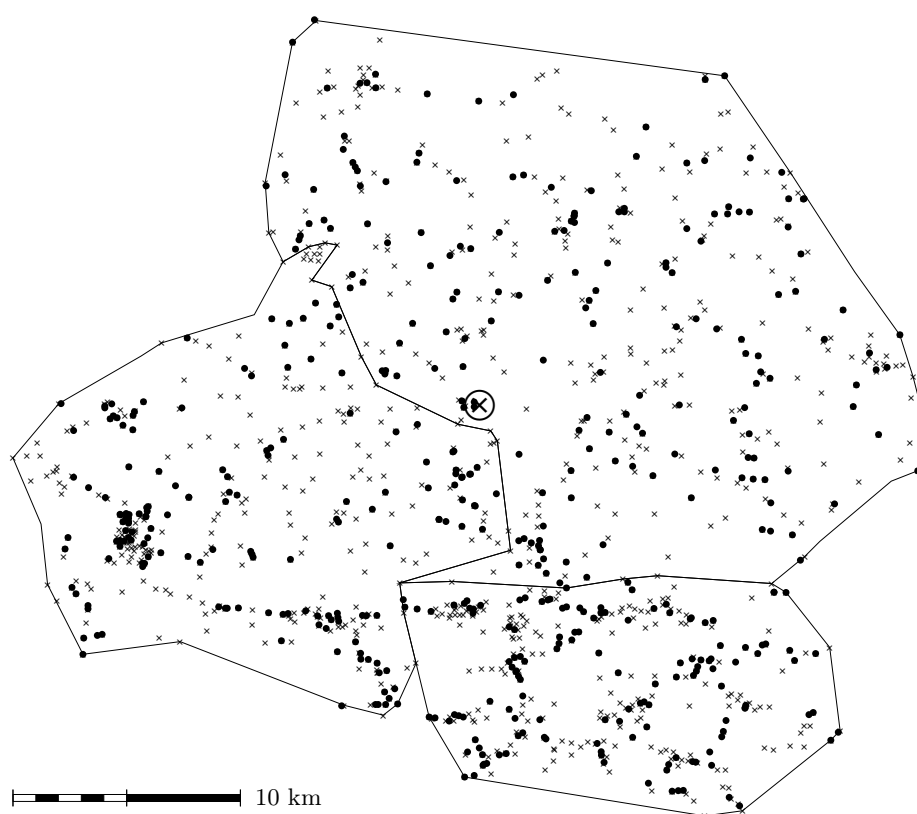
Task	Qualification				Precedence					Duration (minutes)
	Maintenance	Electrician	Earthwork	Foreman	Station Maintenance 3 Lines	Station Maintenance 2 Lines	Switch Service	Cable Repair	Single Task Processes	
off1	x				1	1	1	1		15
off2	x				1	1	1	1		15
off3	x				1					15
srv_station	x	x			2	2				120
on1	x				3	3	2	5		15
on2	x				3	3	2	5		15
on3	x				3					15
excavate			x					2		60
work		x						3		120
backfill			x					4		45
check	x	x							1	20
measure	x	x							1	30
activate		x		x					1	60
supervise		x		x					1	60
inspect	x	x							1	30
srv_site		x							1	60

Table 5.3: Tasks' required qualifications, precedences, and durations

cases was chosen such that it matches the typical workload of ENSO for five working days. Table 5.4 shows the initial set of cases. The assets of the cases were randomly chosen so that the actual distribution and density of the assets in each region is reflected and that the cases could really happen with respect to the network structure of the regions. See Figure 5.4 for illustration.

Process	East	South	West	Total
Station Maintenance 3 Lines	2	2	2	6
Station Maintenance 2 Lines	4	3	4	11
Switch Service	16	12	15	43
Cable Repair	14	11	13	38
Equipment Quick Check	41	32	37	110
Measuring	6	5	6	17
Equipment Activation	3	2	3	8
Construction Supervision	6	5	6	17
Equipment Inspection	7	5	7	19
Customer Site Service	23	18	21	62
Total	122	95	114	331

Table 5.4: Processes and case numbers

Figure 5.4: Task locations ( $\bullet$ ) of simulated cases, depot ( $\otimes$ ), and assets ( $\times$ )

## 5.2 Workforce Scheduling Methods

Methods for generating start solutions of the MWSP-MP were introduced in Section 4.6. The methods are Breadth First Search (BFS) and Depth First Search (DFS). In this section different setups of the scheduling methods are compared by running a scheduler [117] with both methods. As subject of study the region “South” was selected, since it has the fewest cases to be scheduled (see Table 5.4); and thus, the runtime of the scheduler is expected to be the lowest of all regions. The cases are put into the scheduler in the same random order for all scheduler runs. One scheduler run considers the full set of cases and creates the schedules for one working day. The scheduler run ends after all schedules are full. Cases left over due to full schedules of the workers are omitted.

The depot was set to the center of gravity of the region’s assets to avoid excessive influence of the unusual location of the depot. This setup is a result of applying the criterion LD1—introduced in Section 2.5.4—to the region “South”. Table 5.5 illustrates the costs configured for the different optimization objectives (see Section 2.5). Note that the costs reflect the real situation of ENSO, while no reliable estimation could be determined for the profit generated per case. This is a clear difference to the typical logistics problems related to the pickup and delivery of goods at customer sites. The reason for this is the nature of ENSO’s business, which is primarily driven by quality of service considerations being reflected by the number of workers employed. The accounting department thus never calculates profit per case. As a workaround, the profit was set to 100 per task for this work.

Parameter	Value
Travel cost per hour	50
Travel speed on direct line	30 km/h
Waiting cost per hour	21
Case downtime costs per hour	36
Profit	100 per task
Duration of working day	max. 9 hours
Duration of daily break per worker	30 minutes
Time window of daily break per worker	2 hours at middle of working day

Table 5.5: Costs for optimization objectives

To determine the influence of cost considerations as well as of the preliminary sorting of cases, six experiments were performed with the properties depicted in Table 5.6. Since cases with a large number of subsequent tasks exist, it is likely that no feasible insertion of such cases can be found unless those cases are scheduled first. For the experiments with sorted cases, the cases are



sorted by the number of tasks in descending order. It is expected that fewer cases are scheduled in the experiments with sorting (experiments 1, 2, 3).

	Experiment no.					
	1	2	3	4	5	6
Downtime costs	no	no	yes	no	no	yes
Profit	no	yes	yes	no	yes	yes
Cases	sorted			unsorted		

Table 5.6: Properties of scheduling experiments

The method DFS can be performed in three different ways:

- CTR: cases incremental, tasks recursive
- CRTI: cases recursive, tasks incremental
- CRTR: cases recursive, tasks recursive

Due to the large number of 95 cases to be scheduled and the exponentially growing runtime, scheduling cases recursively is not likely to terminate in acceptable time, even if just two solutions are kept per level. Thus, the method DFS will be performed such that the cases are handled incrementally while the tasks of the cases are inserted recursively.

For both methods all experiments are performed for 1, 5, 10, 20, 50, and 100 solutions kept per level. It is expected that the profit increases with an increasing number of solutions kept per level since more solutions are created during the experiment.

All computations were run on a machine with an Intel Core 2 Duo CPU at 2.4 GHz and 2 GB RAM.

**Discussion** Tables 5.7 and 5.8 show the results of the scheduling experiments. Negative profit values indicate that either no profit is added for scheduled cases and tasks (experiments 1 and 4) or that the costs consume the profit added (experiments 3 and 6). As expected, the number of cases scheduled is constantly higher for the experiments with unsorted cases (4, 5, and 6) than for the experiments with sorted cases (1, 2, and 3). It can further be stated that the profit tends to increase with the number of solutions to be kept per level. Exceptions are the experiments 3 and 6 for DFS (see Table 5.8). The reason for this could be attributed to the increasing number of cases for higher numbers of solutions to be kept, since more cases generate a higher amount of travel effort, waiting times, and downtime costs not com-

		Solutions to be kept per level					
		1	5	10	20	50	100
Profit in experiment no.	1	-420	-396	-383	-393	-383	-378
	2	8979	9014	8718	9140	9284	9215
	3	-1460	-2940	-3080	-932	577	121
	4	-615	-549	-508	-477	-480	-510
	5	9184	9150	8991	8991	9387	9488
	6	235	425	-1216	-1302	-1436	276
Cases in experiment no.	1	57	53	53	53	53	47
	2	57	53	50	48	47	49
	3	52	60	61	53	47	50
	4	70	57	66	64	67	64
	5	70	57	61	61	65	66
	6	60	60	69	69	70	63
Runtime (sec) in experiment no.	1	2	16	47	148	719	2239
	2	3	17	49	173	932	2191
	3	3	19	68	182	775	2436
	4	2	18	45	135	810	1810
	5	2	18	42	137	684	2014
	6	2	23	79	271	1232	3048

Table 5.7: Scheduling results for Breadth First Search

		Solutions to be kept per level					
		1	5	10	20	50	100
Profit in experiment no.	1	-420	-517	-487	-533	-538	-520
	2	8979	9395	9506	9761	9761	9588
	3	-1460	-2366	-2336	-3120	-2775	-2474
	4	-615	-544	-567	-588	-544	-544
	5	9184	9455	9455	9455	9455	9455
	6	253	-1850	-1104	-1104	-1914	-1914
Cases in experiment no.	1	57	61	70	77	76	66
	2	57	64	71	76	76	67
	3	52	60	67	70	68	68
	4	70	74	80	80	74	74
	5	70	74	74	74	74	74
	6	60	72	77	77	79	79
Runtime (sec) in experiment no.	1	2	247	2240	1656	4060	13730
	2	2	232	2245	1488	3879	12413
	3	3	803	10049	19042	53293	104653
	4	2	629	197	236	4035	8183
	5	2	454	384	1646	4013	8196
	6	2	992	11820	4322	5073	9877

Table 5.8: Scheduling results for Depth First Search

pensated by the profit. These results are owed to the greed of the algorithms. As soon as a case can be inserted it will be inserted, regardless if a better solution could be found by omitting the case. For a case inserted into almost complete solutions higher travel efforts, waiting times, and downtime costs have to be accepted by the scheduler than for a case inserted into almost empty solutions. Improvements of this behavior are subject to applying more sophisticated scheduling methods like, e.g., meta-heuristics utilizing the neighborhood operators introduced in Sections 4.6.2 and 4.6.3.

The runtimes are significantly higher for DFS than for BFS. The reason is the larger number of task insertions in DFS, since the solution tree is cut only after all tasks of a case were inserted. Extreme growth of runtimes (see experiments 3 and 6 in Table 5.8) results from cases with a large number of tasks to be scheduled early. The number of solutions to be kept per level is a maximum, because fewer feasible solutions may exist for the insertion of a case. If, e.g., a case with 7 tasks is scheduled first and 50 solutions are to be kept, these 50 solutions have 400 positions for the next case's tasks to be inserted. If the next case also contains a large number of tasks, many insertion operations and feasibility checks have to be performed. If, furthermore, no feasible insertion can be found for the second case, the same situation occurs with the third case and so on. Obviously, the downtime costs are chosen such that this situation occurs in the experiments 3 and 6.

As an insight of the experiments, it can be stated that the costs for travel, waiting, downtime, and time window violation need to be fine-tuned as well as the profit per case needs to be fine-tuned to reflect the real situation in BPR projects. It can further be stated that both algorithms behave as expected: the profit tends to raise with more solutions to be kept.

## 5.3 Simulation of Scenarios

### 5.3.1 Method

To check the validity of the CPN models introduced in Sections 3.2 and 3.3, simulations of different scenarios created from ENSO's data were performed. For every scenario the impact on the process execution costs was assumed and compared to the actual simulation results. If the simulation results confirm the assumption with notable statistical significance, the model was considered valid. Changes and their significance were identified by performing the Wilcoxon signed-rank test (see, e.g., [52]). The null hypothesis of the Wilcoxon test assumes that the samples belong to the same basic population. The alternative hypothesis assumes a shift of the values. If the null hypothesis was not accepted, the expected improvement gained by a scenario

was estimated with the estimator for the Hodges-Lehmann nonparametric difference (see, e.g., [52]).

Different parametrizations of the data were created with respect to a selection of the parametrization criteria introduced in Section 2.5.4. To allow for the comparability of the results, the same scheduling method was applied to all simulations. Considering the high and diverging runtimes of the Depth First Search algorithm in Section 5.2, Breadth First Search was chosen for the scheduling of the cases with the parameters listed in Table 5.9. To execute all cases the scheduler runs more than once per simulation, reflecting subsequent working days. Thus, the cases are sorted to consider their priorities.

Parameter	Value
Scheduling Method	Breadth First Search
Solutions to be kept per level	10
Time window handling	strict enforcement
Travel cost per hour	50
Travel speed on direct line	30 km/h
Waiting cost per hour	21
Case downtime costs per hour	360
Profit	100 per task
Case sorting	sorted
Duration of working day	max. 9 hours
Duration of daily break per worker	30 minutes
Time window of daily break per worker	2 hours at middle of working day

Table 5.9: Parameters of the scheduling method

### 5.3.2 Scenarios

The initial result—basis for all comparisons of results—was gained from the simulation of the current situation described in Section 5.1. This situation is referred to as scenario 1 in the following. The costs as introduced in Section 2.5 are generated by workers’ travel times and waiting times. Thus, the criteria chosen for judging a scenario’s quality are travel time per case and waiting time per case. The total finish time for the execution of all cases was chosen as a third criterion, since it serves as an indicator for the distribution of the workload between different regions. If the finishing times of two regions differ, it is likely that the configurations of the regions differ and that the workforce is not utilized optimally. A unified region “All” to be used in the different scenarios was created by merging the regions “East”, “West”, and “South”. This unification is one possible application of the parametrization criterion “Total Number of Regions (TNR)” to the set of regions. Every sce-

nario was simulated 100 times to gain a sufficient sample size. The following scenarios were created:

1. Current situation as introduced in Section 5.1
2. The depots of the regions were set to the respective center of gravity of the regions' assets (criterion LD1 – Location of the Depot), as depicted in Figure 5.5. For the unified region the same centers of gravity were used as for the separated regions. Thus, the unified region has three depots.
3. The qualification of the workers was extended such that every worker could perform every task (criterion AQ – Additional Qualifications). The depots were chosen as for the scenario 1.
4. Combination of scenarios 2 and 3: The depots were chosen as the respective centers of gravity and the qualifications were extended (criteria LD1 and AQ).
5. The assets were assigned to regions such that each asset is assigned to the closest depot (criterion CD – Closest Depot), as depicted in Figure 5.6 (p. 156). The locations of the depots were taken from scenario 2. Since the regions were changed, the evaluation does not consider the separated regions individually but their overall costs compared to the unified region.
6. The scenarios 1 – 4 were simulated with both separated regions and one unified region by applying the criterion TNR – Total Number of Regions.

### 5.3.3 Simulation Results and Discussions

For each region and each scenario it is analyzed if and how the total costs of the scenario change compared to scenario 1. In addition, for each scenario it is compared, whether the execution of all cases is performed cheaper in separated or unified regions. The results are presented for each region as tables and boxplots providing a comparison of the outcomes of the scenarios 1 – 5. Scenario 1 serves as reference for all improvement efforts, since it represents the current situation of ENSO. For each scenario an expectation of the results is provided and the results are discussed. The detailed results are depicted in Tables 5.15, 5.16, 5.17, and 5.18 (pp. 159). Each of the Tables 5.15, 5.16, 5.17, and 5.18 represents the results of the scenarios 1 – 5 for one of the regions.

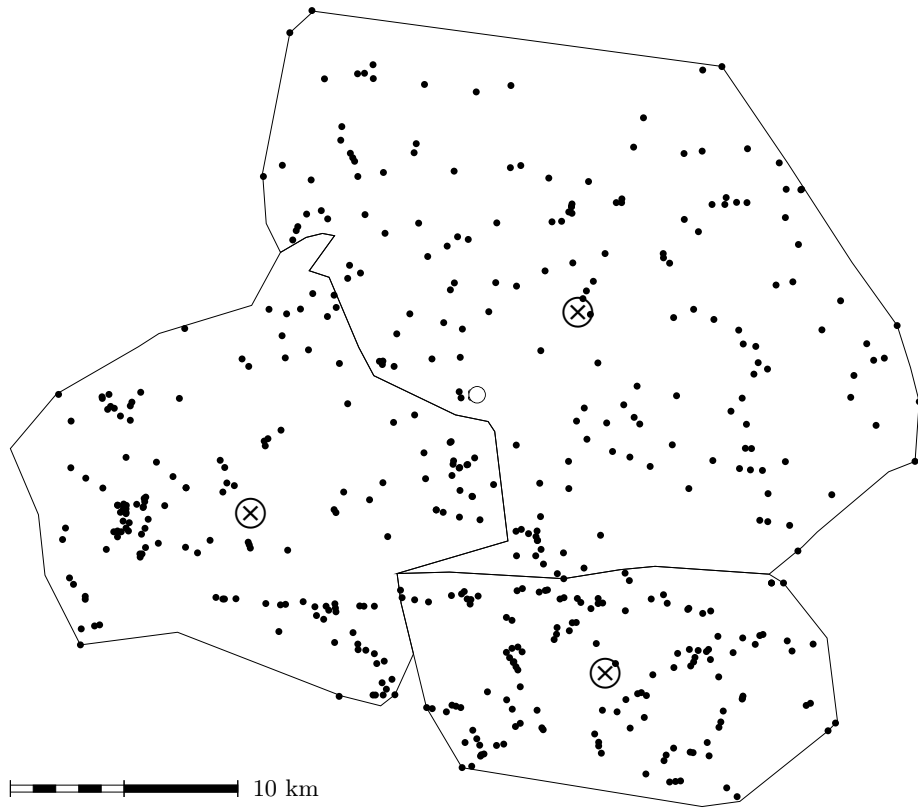


Figure 5.5: Scenarios 2, 4: task locations ( $\bullet$ ), initial depot ( $\circ$ ), and relocated depots ( $\otimes$ )

### 5.3.3.1 Scenario 2: Location of the Depot

The location of the depots is modified such that each depot is at the center of gravity of its region's assets. Figure 5.5 illustrates the change. The center of gravity for the unified region "All" is very close to the current depot. It can thus be assumed, that the simulation results are very close to the ones for scenario 1. Thus, the new depots of the three small regions were taken for the simulations of the region "All", too.

According to the change of the depots' locations depicted in Figure 5.5, decreasing travel effort is expected with the most significant change in the region "South" and the least significant change in the region "East". Since the finishing time of the process execution for all cases corresponds with the travel effort, this value should decrease, too. For the amount of time waited per case no result was assumed preliminarily.

		Region			
		East	South	West	All
Travel per Case	Wilcoxon p-Value	<0.001	≪0.001	≪0.001	≪0.001
	HL difference	45s	529s	397s	329s
Wait per Case	Wilcoxon p-Value	0.085	0.241	0.003	<0.001
	HL difference	—	—	24s	26s
Finish	Wilcoxon p-Value	0.047	≪0.001	≪0.001	≪0.001
	HL difference	1294s	2197s	4895s	2615s

Table 5.10: Statistical analysis: comparison of scenarios 1 and 2

**Discussion** The detailed results—depicted in Tables 5.15, 5.16, 5.17, and 5.18—point at a reduction of the travel effort in all regions. Consider especially the boxplots of the travel costs for the regions “South” (Table 5.16), “West” (Table 5.17), and “All” (Table 5.18). Table 5.10 shows the results of the statistical analysis and the estimated improvements gained with scenario 2. For the region “East” the improvement of travel per case amounts for only 45 seconds, which cannot be read easily from the boxplot of Table 5.15. For the amount of time waited per case Table 5.10 shows a marginal improvement in the regions “West” and “All” while the changes for “East” and “South” are not significant. Considering the results, it can be stated that the travel effort and the finishing time decreased as expected.

### 5.3.3.2 Scenario 3: Qualification of the Workers

In scenario 1 the four different qualifications (see Table 5.3) are distributed among the workers such that “Maintenance” is held by the workers only, “Foreman” is held by the foremen only, and “Electrician” as well as “Earthwork” are held by both groups. The qualification “Foreman” is especially rare, since there are only two foremen per region. The cost of case duration forces the scheduler to prefer solutions with low case durations. This causes the scheduler to accept higher amounts of travel to assign tasks to adequately skilled workers, since it cannot assign tasks to workers without that certain skill, regardless if they are located closer. Additionally, the scheduler may accept waiting times to avoid travel since travel is more expensive than waiting. It is thus likely that two closely located tasks requiring a rare qualification will be executed subsequently by the same worker holding this qualification even though the process structure or time windows may require the worker to wait before the execution of the second of these tasks.

In scenario 3 all workers were assigned with all qualifications. The depots are located at the common central position, as introduced in scenario 1. We expect the travel effort as well as the waiting time to decrease in scenario 3.

**Discussion** The detailed results—depicted in Tables 5.15, 5.16, 5.17, and 5.18—point at a reduction of the travel effort and the waiting time in all regions. Considering the boxplots it can easily be seen, that both travel and waiting times are significantly lower in scenario 3 than in scenario 1. The statistical analysis depicted in Table 5.11 fortifies this appraisal. For all three criteria the Hodges-Lehmann estimator shows considerable improvements in all regions. It can further be stated that in the separated regions (Tables 5.15, 5.16, and 5.17) the median of the waiting time per case of scenario 3 has the lowest value of all scenarios.

		Region			
		East	South	West	All
Travel per Case	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	146s	48s	112s	228s
Wait per Case	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	43s	44s	80s	77s
Finish	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	16845s	1344s	2626s	2240s

Table 5.11: Statistical analysis: comparison of scenarios 1 and 3

### 5.3.3.3 Scenario 4: Depot Location and Qualification of the Workers

Scenario 4 is a combination of scenarios 2 and 3. The depots are set to the centers of gravity of the respective regions, and all workers are assigned with all qualifications. The results of scenario 4 are compared to all of the scenarios 1, 2, and 3. It is expected that the travel effort drops compared to the scenarios 1, 2, and 3. Considering that in scenario 2 the depots are already set to the respective centers of gravity, scenario 4 is expected to improve the travel effort only slightly compared to scenario 2, while the improvement is expected to be considerable compared to scenarios 1 and 3. The waiting times are expected to decrease compared to the scenarios 1, 2, and 3 with the smallest improvement compared to scenario 3.

**Discussion** The detailed results—depicted in Tables 5.15, 5.16, 5.17, and 5.18—point at an overall reduction of the travel effort. Considering the boxplots it can easily be seen, that the travel effort drops least compared to scenario 2. The only exception is region “East”, where the low density of assets and the small displacement of the depot seem to favor scenario 3 over scenario 2 regarding travel effort. The waiting times decrease only compared to scenarios 1 and 2. The inconsistent change of waiting times compared to



scenario 3 is probably due to the already low waiting times in scenario 3 and due to the scheduler preferring less travel effort over less waiting based on their respective costs. The statistical analysis depicted in Table 5.12 fortifies these appraisals.

			Region			
			East	South	West	All
1	Travel per Case	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	199s	558s	494s	524s
	Wait per Case	Wilcoxon p-Value	$\ll 0.001$	0.127	$\ll 0.001$	$\ll 0.001$
		HL difference	39s	—	78s	92s
	Finish	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	17099s	17034s	20246s	4251s
2	Travel per Case	Wilcoxon p-Value	$\ll 0.001$	$< 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	153s	29s	97s	193s
	Wait per Case	Wilcoxon p-Value	$\ll 0.001$	0.006	$\ll 0.001$	$\ll 0.001$
		HL difference	49s	20s	55s	64s
	Finish	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	15589s	14927s	15127s	1645s
3	Travel per Case	Wilcoxon p-Value	$< 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	52s	511s	383s	292s
	Wait per Case	Wilcoxon p-Value	0.414	$\ll 0.001$	0.558	0.039
		HL difference	—	-33s	—	11s
	Finish	Wilcoxon p-Value	0.690	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
		HL difference	—	15722s	17657s	2083s

Table 5.12: Statistical analysis: comparison of scenarios 1 and 4, 2 and 4, 3 and 4

#### 5.3.3.4 Scenario 5: Closest Depot

Scenario 5 is closely related to scenario 2 (Location of Depot). Based on scenario 2 the assets were assigned to the regions “East”, “South”, and “West” such that each asset is closer to the depot of the respective region than to any other depot. The actual operation is a Dirichlet tessellation (or Voronoi decomposition) [85] of the plane, with the depots being the sites of the Dirichlet tessellation and the assets being assigned to the sites, respectively. Figure 5.6 illustrates the resulting regions.

Due to the different assignments of assets to the regions, the number of cases differs from the original ones:

- East: 113 cases (122 before)
- South: 112 cases (95 before)
- West: 106 cases (114 before)

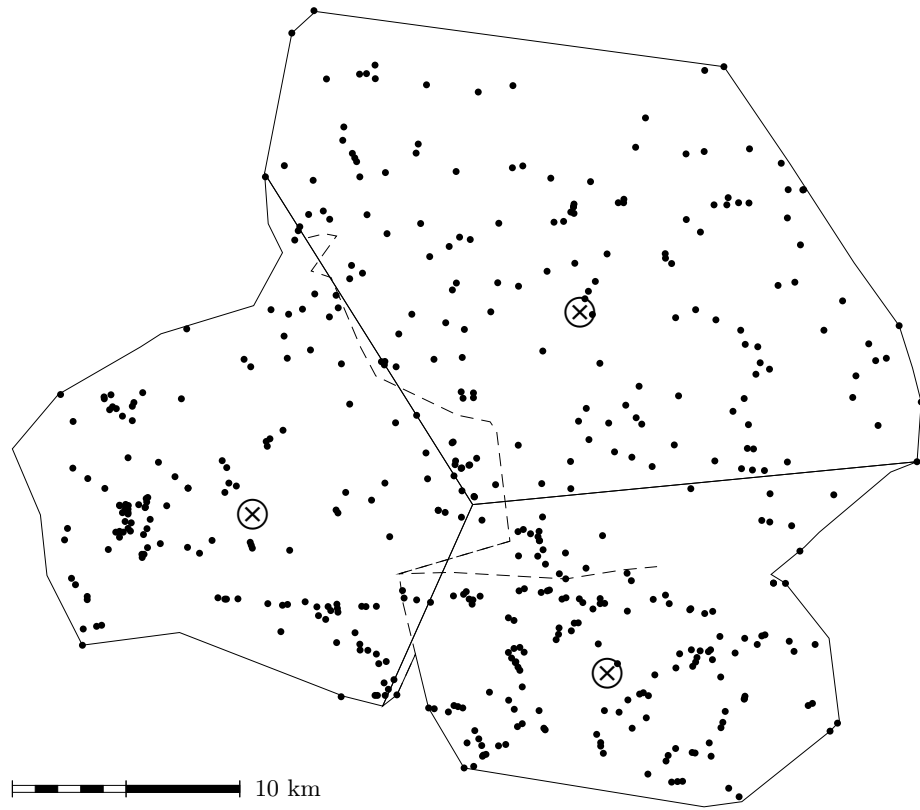


Figure 5.6: Scenario 5: borders as result of Dirichlet tessellation

Nonetheless, since the efficiency of a scenario is determined by the ratios travel per case and wait per case, the regions remain comparable. Considering the number of cases, the total finishing time should drop for the regions “East” and “West” and raise for the region “South”. Considering the borders in Figure 5.6 the biggest change applies to the regions “East” and “South” with “East” getting considerably smaller while “South” becomes larger. The travel effort should thus drop in “East” and raise in “South”. The unified region “All” was not simulated, since reassigning the assets does not impact that region compared to scenario 2.

		Region		
		East	South	West
Travel per Case	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	0.521
	HL difference	180s	-122s	—
Wait per Case	Wilcoxon p-Value	$\ll 0.001$	0.136	0.491
	HL difference	-56s	—	—
Finish	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	18277s	-16129s	5322s

Table 5.13: Statistical analysis: comparison of scenarios 2 and 5

**Discussion** The detailed results—depicted in Tables 5.15, 5.16, 5.17, and 5.18—point at travel effort and finishing times as expected. Considering the boxplots comparing the scenarios 2 and 5 it can easily be seen that the finishing time decreases in the region “East” and raises in the region “South”, while the drop in the region “West” can only be determined to be about 1.5 hours by the Hodges-Lehmann estimator in Table 5.13. Furthermore, Table 5.13 indicates that for the regions “East” and “South” the travel effort changes as expected while for the region “West” no significant change could be achieved. This lack of significance is due to the small change of the region’s setup by the criterion Closest Depot (see Figure 5.6).

### 5.3.3.5 Scenario 6: Total Number of Regions

In the scenarios 1 – 4 results are always compared region-wise. Scenario 6 by contrast compares the combination of the results of the regions “East”, “South”, and “West” to the results of the unified region “All”. This comparison is performed for all criteria applied in the scenarios 1 – 4. Scenario 6 can thus be seen as a meta-scenario. The Wilcoxon test allows for analyzing samples of different size. In scenario 6 the first (or left) sample of the Wilcoxon test was set to be the concatenation of the results of the separated regions. The second (or right) sample of the Wilcoxon test was set to be the results of the unified region. The samples were equally configured for the scenarios 1 – 4. Table 5.14 shows the results of the analysis.

In the unified region workers may travel to locations far away from their depot. It is thus expected that the travel costs are higher for the unified region than for the separated ones. Accordingly, the finish time is also expected to be higher for the unified region.

**Discussion** The results indicate that for all scenarios 1 – 4 (i) the difference between the samples is significant and (ii) the execution in separate regions has cost advantages (negative difference of the Hodges-Lehmann estimator

		compared to scenario			
		1	2	3	4
Travel per Case	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	-252s	-325s	-122s	-229s
Wait per Case	Wilcoxon p-Value	$\ll 0.001$	0.02	0.003	0.039
	HL difference	-35s	-13s	-15s	10s
Finish	Wilcoxon p-Value	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$	$\ll 0.001$
	HL difference	-8806s	-10579s	-9082s	-24554s

Table 5.14: Statistical analysis: separate vs. unified regions in scenario 6

in Table 5.14). The only exception is the waiting time per case compared to scenario 4, where the null hypothesis could neither be accepted nor rejected. This exception is probably for the same reason the waiting times don't drop from scenario 3 to 4: already low values and the scheduler preferring travel over waiting. The reason for the separated regions to perform better is—besides the potentially increased travel effort—probably the distinct number of cases to be scheduled per day. For the unified region the scheduler inserts about three times as much cases as for the separate regions. Thus, it generates a larger number of solutions to choose from per iteration. Nonetheless, the same number of 10 solutions per iteration is chosen as basis for the next insertion. It is thus more likely that the best final solutions are not chosen in the early iterations.

### 5.3.4 Section Summary

Comparing the results of the simulations and the anticipations of possible improvements, the model behaves as expected. The parametrization criteria applied to the mobile environment promise a drop in travel and waiting costs of up to 30%. The magnitude of the improvement may be founded in the unusual location of the original depots, but even when comparing the scenarios with relocated depots (2 and 4, see Table 5.12), the change is significant.

Larger regions tend to have worse results—probably due to the number of solutions chosen by the scheduler per iteration. Thus, in larger regions more solutions should be chosen for the next iteration. This approach causes increasing scheduler execution times for two reasons: (i) due to the larger number of cases in larger regions with more insertions per iteration and (ii) due to the larger set of solutions to be kept per level.

		Scenario				
		1	2	3	4	5
Travel per Case	min	1898	1818	1728	1553	1629
	quantile 25	2034	1980	1888	1828	1814
	median	2088	2044	1933	1891	1861
	quantile 75	2144	2107	2018	1958	1918
	max	2312	2238	2136	2145	2076
Wait per Case	min	70	58	34	35	40
	quantile 25	104	121	67	68	136
	median	131	143	89	94	202
	quantile 75	161	163	113	120	243
	max	388	234	177	193	378
Finish	min	123595	123963	119555	119021	115146
	quantile 25	137803	127298	122211	122070	118460
	median	140762	139588	123229	123348	120319
	quantile 75	143449	142205	124740	124651	122676
	max	148800	148218	144655	129214	125081

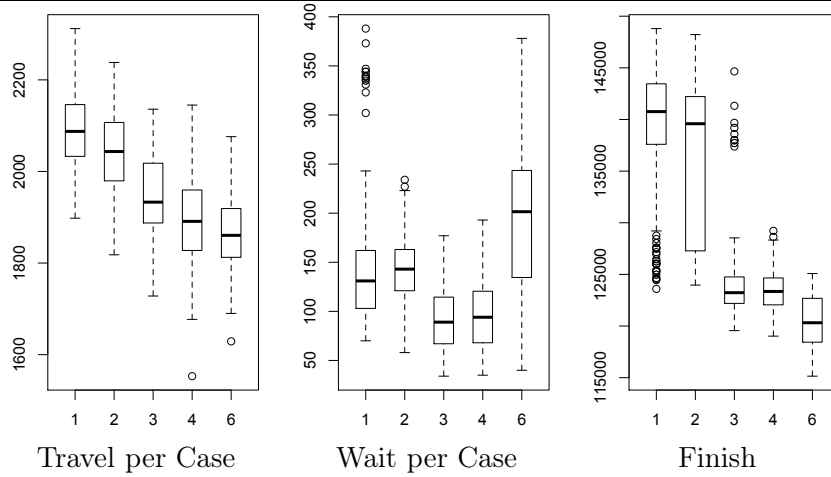


Table 5.15: Simulation results for the region “East” (in seconds)

		Scenario				
		1	2	3	4	5
Travel per Case	min	1575	1022	1551	1054	1189
	quantile 25	1676	1140	1637	1120	1267
	median	1708	1183	1661	1150	1301
	quantile 75	1745	1221	1684	1181	1349
	max	1833	1304	1799	1246	1415
Wait per Case	min	61	63	51	70	54
	quantile 25	123	131	89	122	136
	median	163	164	115	145	184
	quantile 75	197	205	137	173	232
	max	511	356	226	356	492
Finish	min	90968	88226	89510	87717	113550
	quantile 25	105685	103401	104712	89215	118620
	median	106726	104778	105396	89625	120873
	quantile 75	107982	106106	106320	90518	122284
	max	114685	111936	110014	92018	124622

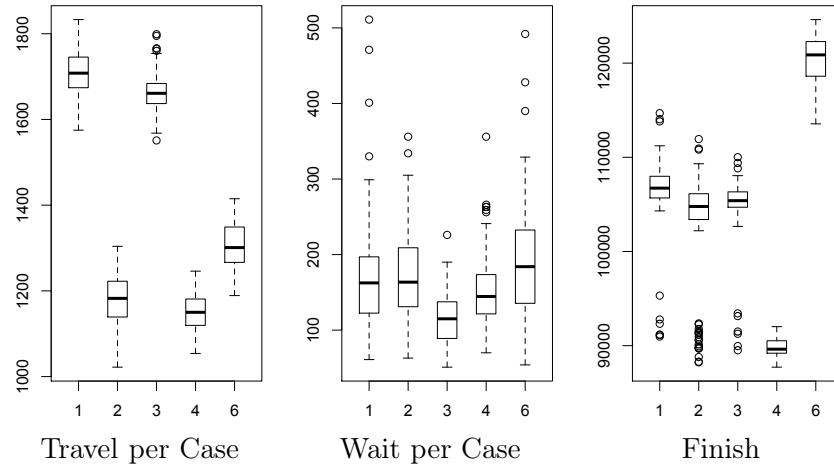


Table 5.16: Simulation results for the region “South” (in seconds)

		Scenario				
		1	2	3	4	5
Travel per Case	min	1692	1337	1643	1243	1362
	quantile 25	1821	1425	1718	1323	1423
	median	1862	1464	1746	1367	1463
	quantile 75	1903	1504	1782	1410	1511
	max	2045	1611	1914	1533	1648
Wait per Case	min	52	47	14	25	60
	quantile 25	111	105	54	58	106
	median	161	139	77	78	139
	quantile 75	204	158	98	104	168
	max	464	255	230	192	268
Finish	min	110091	92985	98276	88045	89941
	quantile 25	113348	98696	111159	93330	93381
	median	115065	110774	112361	94625	105548
	quantile 75	116573	112616	113723	96163	107399
	max	120695	119257	117488	113652	111865

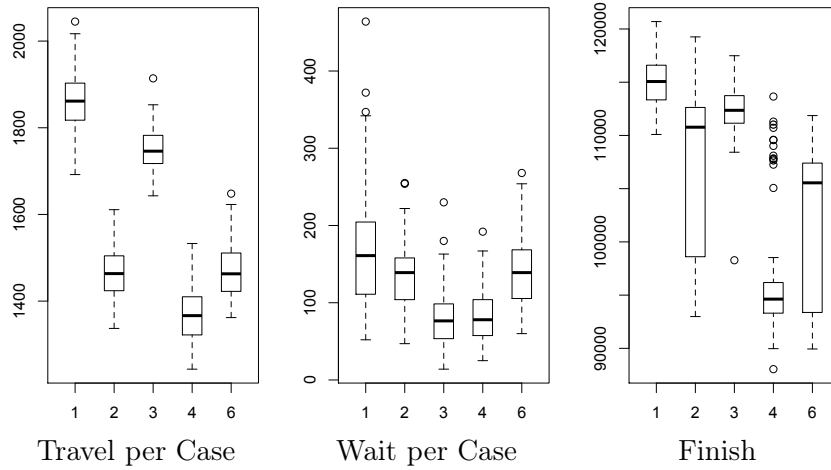


Table 5.17: Simulation results for the region “West” (in seconds)

		Scenario				
		1	2	3	4	5
Travel per Case	min	1925	1574	1745	1494	—
	quantile 25	2088	1746	1826	1565	—
	median	2118	1789	1888	1587	—
	quantile 75	2146	1830	1941	1621	—
	max	2255	1920	2019	1727	—
Wait per Case	min	99	75	36	40	—
	quantile 25	154	129	73	66	—
	median	183	162	106	81	—
	quantile 75	223	192	149	138	—
	max	336	269	233	199	—
Finish	min	118730	115883	115852	115576	—
	quantile 25	122155	119758	120200	117480	—
	median	123671	121291	121198	119625	—
	quantile 75	124941	123309	123390	121276	—
	max	141711	125862	127083	125722	—

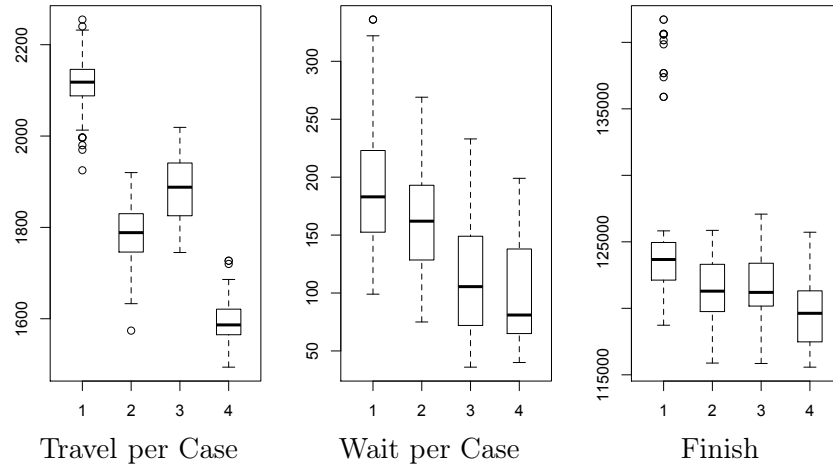


Table 5.18: Simulation results for the unified region (in seconds)



## 5.4 Tool Support

To validate the method, a tool—Mobile Organization Sim Control (MoSim)—was developed with the help of Sebastian Neudert in [84] and Steffen Schulz in [104] to support the creation of scenarios. The tool is an Eclipse Rich Client Platform<sup>1</sup> application. As already introduced in Chapter 3, CPN Tools is used to run the actual simulations. The tool provides an interface to create the scenarios and to model the mobile business processes for execution in CPN Tools.

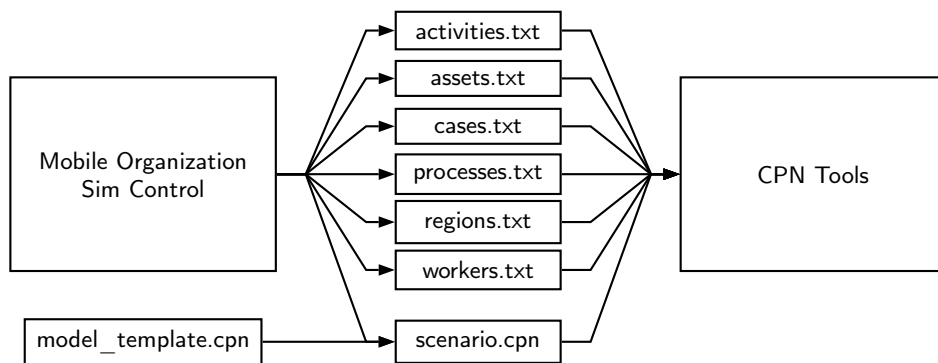


Figure 5.7: Tool Relations

### 5.4.1 Manipulating Simulation Data

The scenarios as introduced above are the input data to the actual CPN model. They consist of processes, assets, activities, cases, processes, regions, and workers. CPN Tools expects the input data as CPN ML variables in text files in a fixed file system structure. The tool is able to read and write such files in this structure. Figure 5.7 illustrates the files written by the MoSim tool and read by CPN Tools. Figure 5.8 gives an impression of the user interface.

The logical structure of the simulation data follows the domain model entities with assets and workers assigned to regions. The tool obeys this structure building a hierarchy of objects (see the left navigation view in Figure 5.8). The entities are displayed in a map view, with information about attributes available on click. All data can be manipulated in property panes of the objects. To change the assignment of workers and assets to regions, they can simply be dragged from one region to another. A dragged entity remains at the same geographical position, but the software recalculates the region view and updates the internal object hierarchy (see Figures 5.9 and 5.10

<sup>1</sup><http://www.eclipse.org/home/categories/rcp.php>

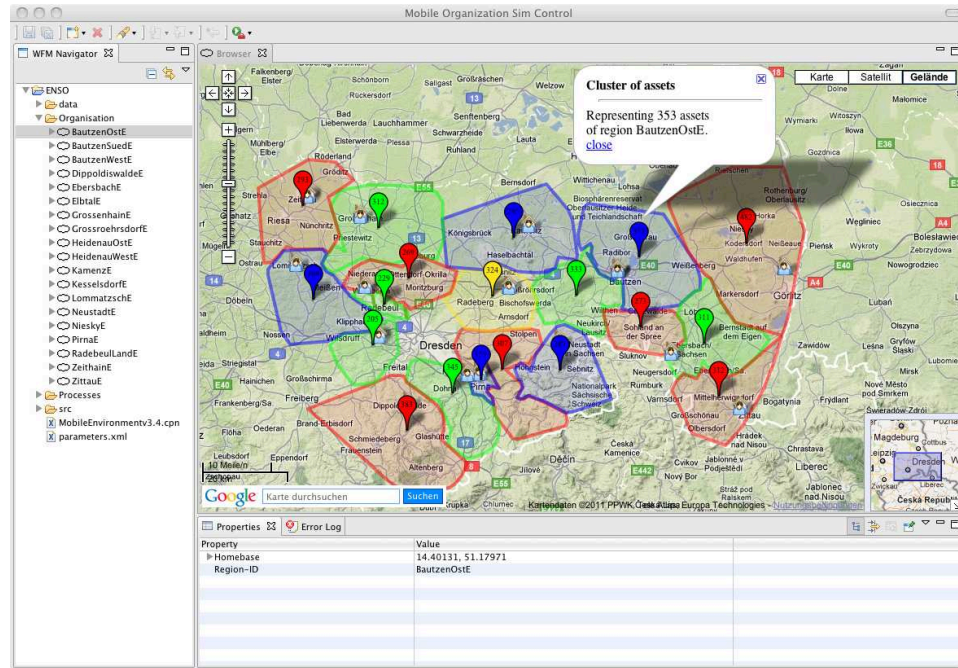


Figure 5.8: Tool overview

for illustration, where the marked asset was dragged into the region on the right to change its assignment). The resulting scenarios can be stored in different projects for different simulation runs. Future versions of the software will contain functions to generate the scenarios directly by selecting parametrization criteria from the catalog introduced in Section 2.5.4.

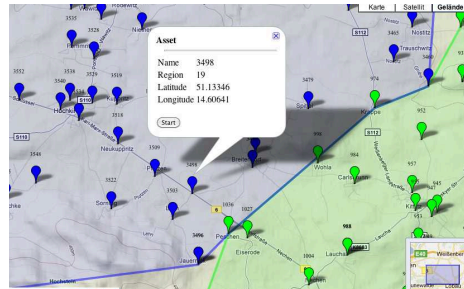


Figure 5.9: Before asset drag-and-drop

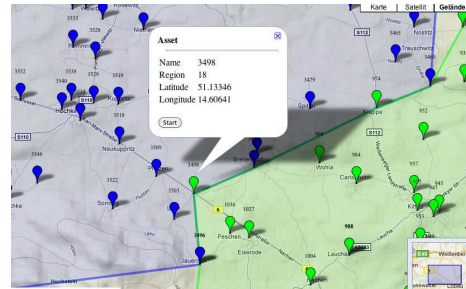


Figure 5.10: After asset drag-and-drop

### 5.4.2 Modeling Business Processes

As already introduced in Chapter 3, business processes become an integral part of the CPN model to be executed. Every process is represented by

a CPN which is interwoven with the static model elements. The MoSim tool contains a graphical SMPL process editor. To keep the process models manageable they can be provided as SMPL models and are transformed into CPNs. The transformation of SMPL models to CPN process models is performed left-to-right from the InitialNode to the FinishNode of the SMPL model, applying the rules introduced in Section 3.4.4. Figure 5.11 shows a screen of the software with a cutout of the generated CPN. Note that the generated CPN is available to the user only for information reasons. A user usually does not need to interact with this model.

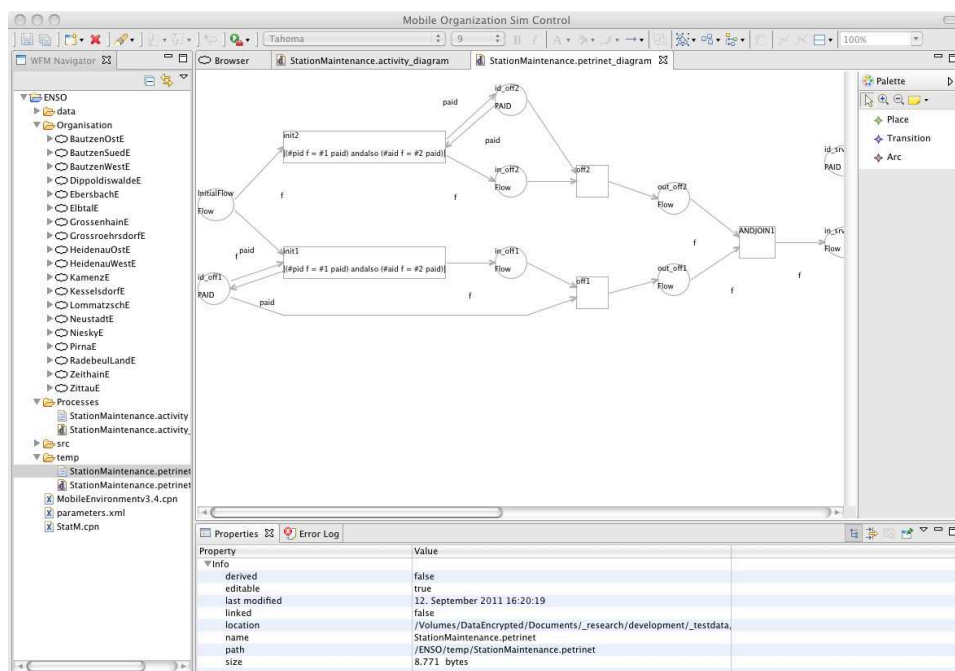


Figure 5.11: Generated CPN process model

Since the processes are integral parts of the CPN model, they cannot be loaded by CPN Tools at simulation runtime. Instead, the MoSim tool refers to a CPN model template with the static part of the domain model (see Section 3.2). CPN model files are XML-files into which the business processes can be inserted in a structured way. The CPNs generated from the SMPL model are injected into the template file, building a new CPN model. The resulting CPN model is written into a fixed file system structure readable by CPN Tools. The simulations can then be run by starting CPN Tools and loading the project directory created by the Mobile Organization Sim Control software. The bottom part of Figure 5.7 illustrates the relation between the template file, the MoSim tool, and CPN Tools.

## 5.5 Chapter Summary

In this chapter the CPN domain model and the basic scheduling algorithms were validated. As subject of study the ENSO gas and power supply was introduced and supporting software was presented. The scheduling algorithms behave as expected. Nonetheless, they are just a proof of concept in the sense that Breadth First Search and Depth First Search are simple construction methods. For better results state-of-the-art algorithms like meta-heuristics should be implemented and examined.

The executable CPN domain model was examined by creating parametrized scenarios and simulating them. The simulation results suggest that the model is capable of predicting the outcomes of BPR projects in mobile environments. An important result of the simulation runs is that the cost and profit values need to be fine-tuned according to the real project situation to gain resilient statements about the environment under examination.

## Chapter 6

# Conclusion

### 6.1 Scientific Contributions

This work contributes to business process research in several ways: it identifies the properties of spatially distributed work and formulates an according domain model of mobile work. The domain model is independent of the actual business domain, since it focuses on the mechanisms of task assignment, process execution, and synchronization of remote tasks. Besides providing the fundamental terminology of this work the domain model is a general reference model of mobile work to be referred to in future research. Furthermore, common optimization objectives of mobile business processes are presented, and a set of parametrization criteria for their execution environments is developed. The business independence of the domain model empowers business analysts to optimize aspects that are not directly related to the business objectives. Among these aspects are the distribution of workers, the setup of regions (e. g., the location and number of the depots) as well as the regions' borders and their total number. Mobile processes and execution environments of various business domains can be modeled, parameterized, and simulated. Among the supported domains are utilities, telecommunication providers, home appliances service, and any other domain where workers perform tasks in the field. In Chapter 3 an executable form of the domain model is developed that allows for the simulation of mobile business process sets. The executable form rests upon a formulation in colored Petri nets to provide access to the simulation of complex process landscapes. Since CPNs are not very common to average business users, the Simple Mobile Process Language (SMPL)—a subset of UML activity diagrams—is introduced. The SMPL is easy to learn and the modeled mobile processes can be transformed to CPNs. A scheme for transforming SMPL models to CPN models is given as well as a supporting tool to perform the transformation. Using the tool leaves the resulting CPNs transparent to the user. The validation in Chapter

5 shows, that simulating optimization scenarios of mobile environments with the method and toolset presented is capable of supporting business analysts in predicting the effectiveness of business change projects before the actual change management takes place.

Simulating and executing the business process models requires assigning tasks to workers with respect to the structure of the processes. The Mobile Workforce Scheduling Problem with Multitask Processes (MWSP-MP) was formulated in Chapter 4 to provide basic scheduling algorithms for the domain model. The MWSP-MP is a new generalization of a number of well-known scheduling problems like the Traveling Salesman Problem, the Vehicle Routing Problem, and the Resource Constrained Project Scheduling Problem. Extending the VRP, the MWSP-MP introduces multitask-processes with existence costs, task durations, skill demands, and precedence constraints of the tasks. Extending the RCPSP class of problems it introduces case existence costs and spatially distributed tasks. To our best knowledge the MWSP-MP is the most general formulation of a scheduling problem describing mobile work and multitask processes. The neighborhood operators INSERT and REMOVE for the manipulation of solutions to the MWSP-MP are defined. It has already been shown in [40], that the operators INSERT and REMOVE are sufficient to define further neighborhood operators like SHIFT, EXCHANGE, or REPLACE. Thus, these operators can be utilized to apply sophisticated methods like meta-heuristics to the MWSP-MP. For the validation of the domain model by simulating real world processes two basic scheduling algorithms were implemented: Breadth First Search and Depth First Search.

## 6.2 Future Research

This work demonstrates how mobile multitask processes of a business organization can be modeled and simulated regardless of the business domain. Future research should be conducted for mobile processes performed by multiple organizations. An example for such a process is the damage search and repair process introduced in Section 2.1.3, where the earthwork operations could be executed by a subcontractor of the utility.

The MWSP-MP introduced in this work is capable of considering a variety of real-life requirements. Nonetheless, it is currently not capable of handling task synchronization if the data network breaks down. Real-world schedulers for the MWSP-MP that reschedule the workers if urgent cases occur must be able to handle disconnected clients. This problem is closely related to the rollback of tasks and cases. A scheduler should also be able to perform such operations in the case of disconnected workers and unknown states of cases and tasks. The scheduling algorithms presented in this work are very basic

construction methods. A real-world scheduler would have to be implemented in a more sophisticated way, using improvement methods or meta-heuristics.

Currently, the software presented works in a one-way manner. A user can create scenarios and export them to the simulation tool. Monitors integrated into the model write the simulation results to text files. Analyzing simulation results and creating new scenarios accordingly would be a desirable additional feature of the tool. This would allow for the automated creation and execution of slightly different scenarios without user interaction. It is also desirable that the tool can create scenarios in accordance with the parametrization criteria introduced in Section 2.5.4.





# Symbols

In general symbols are used as follows:

*property*<sub>*object*</sub><sup>*attribute*</sup>

$a$	An activity
$\mathcal{A}$	Set of activities
$c$	A case
$\mathcal{C}$	Set of cases
$\mathcal{CS}$	Set of control structures
$d$	Demand for a resource
$d^{max}$	Maximum capacity
$\mathcal{D}$	Set of depots
$\delta$	Load at a node
$e$	An asset
$\mathcal{E}$	Set of assets
$\mathcal{F}$	Set of control flow edges
$\mathcal{I}$	Set of incidences
$\mathcal{K}$	Set of edges between nodes
$\kappa$	Costs
$\kappa^{dtravel}$	Travel distance costs
$\kappa^{dur}$	Duration costs
$\kappa^{ex}$	Existence costs

$\kappa^q$	Qualification costs
$\kappa^{travel}$	Travel costs
$\kappa^{tt}$	Total time and travel costs
$\kappa^{ttravel}$	Travel time costs
$\kappa^u$	Total costs per time unit
$\kappa^{ub}$	Basic costs per time unit
$\kappa^{ud}$	Travel costs per distance unit
$\kappa^{udur}$	Duration costs per time unit
$\kappa^{uv}$	Time window violation costs per time unit
$\kappa^v$	Time window violation costs
$\kappa^{wait}$	Waiting costs
$\kappa^{work}$	Working costs
$\kappa^y$	Yearly total costs
$l$	A location
$l^{lon}$	Longitudinal component of location $l$
$l^{lat}$	Latitudinal component of location $l$
$l^{depot}$	Location of a depot
$n^u$	Number of time units
$n^{visits}$	Number of visits
$p$	A process
$\mathcal{P}$	Set of processes
$\pi$	Precedence index
$\mathcal{Q}$	Set of qualifications
$q^{req}$	$ \mathcal{Q} $ -Tuple of binary values representing required qualifications
$q^{avail}$	$ \mathcal{Q} $ -Tuple of binary values representing available qualifications
$q_i$	Binary value of the $i$ th qualification
$r$	A region

$\mathcal{R}$	Set of regions
$s$	A solution of the MWSP-MP
$\sigma$	A schedule
$t^{arrive}$	Time stamp of the arrival of a worker
$t^{def}$	Default time
$t^{finish}$	Time stamp of the end of a working period
$t^{max}$	Upper bound of a working period
$t^{min}$	Lower bound of a working period
$t^{start}$	Time stamp of the start of a working period
$t^{travel}$	Travel time
$t^{wait}$	Waiting time
$t^{work}$	Working time
$t_{\tau}^{minins}$	Largest finishing time of task $\tau \in \mathcal{T}$
$t_{\tau}^{push}$	Push forward time of task $\tau \in \mathcal{T}$
$t_{\tau}^{slack}$	Slack time of task $\tau \in \mathcal{T}$
$\tau$	A task
$\tau_{depot}$	Task representing the depot in schedules
$\tau_i$	The $i$ th task of a schedule
$\mathcal{T}$	Set of tasks
$w$	A worker
$\mathcal{W}$	Set of workers



# Bibliography

- [1] E.H.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.
- [2] R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A Survey of Very Large Scale Neighborhood Search Techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.
- [3] S. Anavi-Isakow and B. Golany. Managing multi-project environments through constant work-in-process. *International Journal of Project Management*, 21(1):9–18, 2003. doi:10.1016/s0263-7863(01)00058-8.
- [4] J. Antes and U. Derigs. A new parallel tour construction algorithm for the vehicle routing problem with time windows. Technical report, Department of Information Systems and Operation Research, University of Cologne, Cologne, 1995.
- [5] R. Ash and D.E. Smith-Daniels. The Effects of Learning, Forgetting, and Relearning on Decision Rule Performance in Multiproject Scheduling. *Decision Sciences*, 30(1):47–82, 1999. doi:10.1111/j.1540-5915.1999.tb01601.x.
- [6] R. Baldacci, M. Battarra, and D. Vigo. Routing a Heterogeneous Fleet of Vehicles. In B.L. Golden, S. Raghavan, E.A. Wasil, R. Sharda, and S. Voß, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 3–27. Springer US, 2008. doi:10.1007/978-0-387-77778-8\_1.
- [7] L. Baresi and M. Pezzè. Improving UML with Petri nets. *Electronic Notes in Theoretical Computer Science*, 44(4):107–119, 2001. doi:10.1016/S1571-0661(04)80947-2.
- [8] J. Becker, M. Kugeler, and M. Rosemann, editors. *Prozessmanagement*. Springer, 2008. Available from: <http://www.springer.com/978-3-540-79248-2>.

- [9] G. Berio, K. Mertins, and F.W. Jaekel. Common Enterprise Modelling Framework for Distributed Organisations. In P. Piztek, editor, *Proceedings of the 16th IFAC World Congress*, 2005.
- [10] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003. doi:10.1145/937503.937505.
- [11] D.B. Bock and J.H. Patterson. A Comparison of Due Date Setting, Resource Assignment, and Job Preemption Heuristics for the Multi-project Scheduling Problem. *Decision Sciences*, 21(1):387–402, 1990. doi:10.1111/j.1540-5915.1990.tb00321.x.
- [12] V. Bosilj-Vuksic, J. Jaklič, and A. Popovič. Business Process Change and Simulation Modelling. *Systems Integration*, 29:29–37, 2005.
- [13] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1):104–118, 2005. doi:10.1287/trsc.1030.0056.
- [14] O. Bräysy and M. Gendreau. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005. doi:10.1287/trsc.1030.0057.
- [15] P. Brucker, A. Drexler, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999. doi:10.1016/S0377-2217(98)00204-5.
- [16] A.M. Campbell and M.W.P. Savelsbergh. Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems. *Transportation Science*, 38(3):369–378, 2004. doi:10.1287/trsc.1030.0046.
- [17] J. Campos and J. Merseguer. On the Integration of UML and Petri Nets in Software Development. In S. Donatelli and P. Thiagarajan, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2006*, volume 4024 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin / Heidelberg, 2006. doi:10.1007/11767589\_2.
- [18] S. Christensen and T.B. Haagh. Design/CPN - Overview of CPN ML Syntax. Technical report, University of Aarhus, 1996.
- [19] G. Clarke and J.W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 1964. doi:10.1287/opre.12.4.568.

- [20] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. VRP with Time Windows. In Toth and Vigo [115], pages 157–193. Available from: <http://dl.acm.org/citation.cfm?id=505847.505854>.
- [21] P. Cowling, N. Colledge, K. Dahal, and S. Remde. The Trade Off Between Diversity and Quality for Multi-objective Workforce Scheduling. In J. Gottlieb and G. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin / Heidelberg, 2006. doi:10.1007/11730095\_2.
- [22] T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.
- [23] G.B. Dantzig and J.H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959. doi:10.1287/mnsc.6.1.80.
- [24] G.J. de Vreede, A. Verbraeck, and D.T.T. van Eijck. Integrating the Conceptualization and Simulation of Business Processes: A Modeling Method and Arena Template. *Simulation*, 79(1):43–55, 2003. doi:10.1177/0037549703254725.
- [25] E. Demeulemeester. *Optimal algorithms for various classes of multiple resource-constrained project scheduling problems*. PhD thesis, KU Leuven, 1992.
- [26] G. Doumeingts, B. Vallespir, M. Zanettin, and D. Chen. GIM: GRAI Integrated Methodology, A Methodology for Designing CIM Systems. Technical report, GRAI/LAP, Université Bordeaux, 1992.
- [27] E.J. Dumond and J. Dumond. An Examination of Resourcing Policies for the Multi-resource Problem. *International Journal of Operations & Production Management*, 13(5):54–75, 1993. doi:10.1108/01443579310028175.
- [28] J. Dumond. In a multi-resource environment, how much is enough? *International Journal of Production Research*, 30(2):395–410, 1992. doi:10.1080/00207549208942902.
- [29] J. Dumond and V.A. Mabert. Evaluating Project Scheduling and Due Date Assignment Procedures: An Experimental Analysis. *Management Science*, 34(1):101–118, 1988. doi:10.1287/mnsc.34.1.101.
- [30] W. Emmerich and V. Gruhn. FUNSOFT Nets: A Petri-Net based Software Process Modeling Language. In *Proceedings of the 6th International Workshop on Software Specification and Design, IWSSD '91*,

- pages 175–184. IEEE Computer Society Press, 1991. doi:10.1109/IWSSD.1991.213063.
- [31] H. Eshuis. *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*. PhD thesis, University of Twente, 2002.
  - [32] B. Fleischmann, S. Gnutzmann, and E. Sandvoß. Dynamic Vehicle Routing Based on Online Traffic Information. *Transportation Science*, 38(4):420–433, 2004. doi:10.1287/trsc.1030.0074.
  - [33] M.S. Fox and M. Gruninger. Enterprise Modeling. *AI Magazine*, 19(3):109–121, 1998. Available from: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.9553>.
  - [34] M. Gaitanides. *Prozessorganisation*. Vah lens Handbücher der Wirtschafts- und Sozialwissenschaften. Verlag Franz Vahlen GmbH, München, 2007. Available from: <http://www.vahlen.de/productview.aspx?product=13437>.
  - [35] J. Garrido and M. Gea. A Coloured Petri Net Formalisation for a UML-Based Notation Applied to Cooperative System Modelling. In P. Forbrig, Q. Limbourg, J. Vanderdonckt, and B. Urban, editors, *Interactive Systems: Design, Specification, and Verification*, volume 2545 of *Lecture Notes in Computer Science*, pages 16–28. Springer Berlin / Heidelberg, 2002. doi:10.1007/3-540-36235-5\_2.
  - [36] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies*, 14(3):157–174, 2006. doi:10.1016/j.trc.2006.03.002.
  - [37] M. Gendreau, F. Guertin, J.-Y. Potvin, and É. Taillard. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, 33(4):381–390, April 1999. doi:10.1287/trsc.33.4.381.
  - [38] M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the Capacitated VRP. In Toth and Vigo [115], pages 129–154. Available from: <http://portal.acm.org/citation.cfm?id=505847.505853>.
  - [39] G.M. Giaglis, R.J. Paul, and V. Hlupic. Integrating simulation in organizational design studies. *International Journal of Information Management*, 19(3):219–236, 1999. doi:10.1016/S0268-4012(99)00015-8.
  - [40] A. Goel. *Fleet Telematics: Real-Time Management and Planning of Commercial Vehicle Operations*. Springer Verlag, 2008. Available from: <http://www.springer.com/978-0-387-75104-7>.



- [41] A. Goel and V. Gruhn. A General Vehicle Routing Problem. *European Journal of Operational Research*, 191(3):650–660, 2008. doi:10.1016/j.ejor.2006.12.065.
- [42] A. Goel, V. Gruhn, and T. Richter. Mobile Workforce Scheduling Problem with Multitask-Processes. In S. Rinderle-Ma, S. Sadiq, F. Leymann, W.M.P. van der Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, and C. Szyperski, editors, *Business Process Management Workshops*, volume 43 of *Lecture Notes in Business Information Processing*, pages 81–91. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-12186-9\_9.
- [43] J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem. *European Journal of Operational Research*, 189(3):1171–1190, 2008. doi:10.1016/j.ejor.2006.06.074.
- [44] R.C. Gronback. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Addison-Wesley Professional, 1 edition, 2009. Available from: <http://www.informit.com/store/product.aspx?isbn=0321534077>.
- [45] Object Management Group. Unified Modeling Language: Superstructure Version 2.0. Technical report, Object Management Group, 2005. Available from: <http://www.omg.org/spec/UML/2.0/>.
- [46] Object Management Group. Business Process Model and Notation (BPMN) Version 2.0. Technical report, Object Management Group, 2010. Available from: <http://www.omg.org/spec/BPMN/2.0/>.
- [47] V. Gruhn, A. Köhler, and R. Klawes. Modeling and analysis of mobile business processes. *Journal of Enterprise Information Management*, 20(6):657–676, 2007. doi:10.1108/17410390710830718.
- [48] V. Gruhn and R. Laue. Reducing the Cognitive Complexity of Business Process Models. In G. Baciú, Y. Wang, Y. Yao, W. Kinsner, K. Chan, and L.A. Zadeh, editors, *Proceedings of the 8th IEEE International Conference on Cognitive Informatics (ICCI '09)*, pages 339–345. IEEE Computer Society, 2009. doi:10.1109/coginf.2009.5250717.
- [49] V. Gruhn and T. Richter. A General Model of Mobile Environments: Simulation Support for Strategic Management Decisions. In *Proceedings of the 7th International Conference on Grid and Cloud Computing*, pages 753–764. IEEE Computer Society, 2008. doi:10.1109/GCC.2008.25.

- [50] V. Gruhn and T. Richter. Two-Stage Process Modeling for Simulation in BPR. In *1st International Conference on Advances in System Simulation (SIMUL'09)*, volume 0, pages 132–137, Los Alamitos, CA, USA, 2009. IEEE Computer Society. doi:10.1109/simul.2009.18.
- [51] V. Gruhn and T. Richter. Business Independent Model of Mobile Workforce Management. In P. Bellavista, R.-S. Chang, H.-C. Chao, S.-F. Lin, and P. Sloot, editors, *Advances in Grid and Pervasive Computing*, volume 6104 of *Lecture Notes in Computer Science*, pages 552–561. Springer Berlin / Heidelberg, 2010. doi:10.1007/978-3-642-13067-0\_57.
- [52] S.S. Gupta and S. Panchapakesan. *Multiple Decision Procedures: Theory and Methodology of Selecting and Ranking Populations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2002. Available from: <http://www.ec-securehost.com/SIAM/CL44.html>.
- [53] P. Hansen and N. Mladenović. A Tutorial on Variable Neighborhood Search. *Les Cahiers du GERAD*, July 2003.
- [54] W.S. Herroelen, B. de Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 25(4):279–302, 1998. doi:10.1016/S0305-0548(97)00055-5.
- [55] K. Jensen and L.M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer-Verlag, Berlin Heidelberg, Germany, 2009. Available from: <http://www.springer.com/978-3-642-00283-0>.
- [56] H.J. Johansson, P. Machugh, A.J. Pendlebury, and W.A. Wheeler. *Business Process Reengineering: Breakpoint Strategies for Market Dominance*. Wiley New York, 1993.
- [57] J.B. Jørgensen and K.B. Lassen. Requirements Engineering for the Adviser Portal Bank System. In *13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*, pages 259–268. IEEE Computer Society, 2006. doi:10.1109/ECBS.2006.60.
- [58] M. Kakihara and C.F. Sørensen. Mobility: An Extended Perspective. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS)*, pages 1756–1766, January 2002. doi:10.1109/hicss.2002.994088.

- [59] C. Kecher. *UML 2 - Das umfassende Handbuch*. Galileo Computing, Bonn, 2009. Available from: <http://www.galileocomputing.de/2134>.
- [60] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "ereignisgesteuerter Prozessketten (EPK)". *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, (89), 1992.
- [61] G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1998.
- [62] A. Khneisseh. *Geschäftsprozessmodellierung mit Petri-Netzen*. Expert-Verlag, 2005.
- [63] P. Kilby, P. Prosser, and P. Shaw. A Comparison of Traditional and Constraint-based Heuristic Methods on Vehicle Routing Problems with Side Constraints. *Constraints*, 5(4):389–414, 2000. doi:10.1023/A:1009808327381.
- [64] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.
- [65] A. Klebeck. Routenoptimierung in mobilen Workflow Management- und Workforce Management Systemen. Master's thesis, Universität Leipzig, Leipzig, November 2007.
- [66] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333, 1996. doi:10.1016/0377-2217(95)00357-6.
- [67] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Management Science*, 41(10):1693–1703, 1995. doi:10.1287/mnsc.41.10.1693.
- [68] K. Kosanke, F. Vernadat, and M. Zelm. CIMOSA: enterprise engineering and integration. *Computers in Industry*, 40(2-3):83–97, 1999. doi:10.1016/S0166-3615(99)00016-0.
- [69] D. Krüger and A. Scholl. A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 197(2):492–508, 2009. doi:10.1016/j.ejor.2008.07.036.

- [70] I. Kurtulus and E.W. Davis. Multi-Project Scheduling: Categorization of Heuristic Rules Performance. *Management Science*, 28(2):161–172, 1982. doi:10.1287/mnsc.28.2.161.
- [71] G. Laporte and F. Semet. Classical Heuristics for the Capacitated VRP. In Toth and Vigo [115], pages 109–128.
- [72] D. Lesaint, C. Voudouris, N. Azarmi, I. Alletson, and B. Laithwaite. Field workforce scheduling. *BT Technology Journal*, 21(4):23–26, 2003. doi:10.1023/A:1027315016892.
- [73] S. Lin. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 44(10):2245–2269, December 1965.
- [74] J.P. López-Grao, J. Merseguer, and J. Campos. From UML Activity Diagrams To Stochastic Petri Nets: Application To Software Performance Engineering. In *Proceedings of the 4th International Workshop on Software and Performance*, WOSP '04, pages 25–36, New York, NY, USA, 2004. ACM. doi:10.1145/974044.974048.
- [75] H.R. Lourenço, O. Martin, and T. Stützle. Iterated Local Search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 320–353. Springer New York, 2003. doi:10.1007/0-306-48056-5\_11.
- [76] Q. Lu and M.M. Dessouky. A New Insertion-based Construction Heuristic for Solving the Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research*, 175(2):672–687, 2006. doi:10.1016/j.ejor.2005.05.012.
- [77] P. Luff and C. Heath. Mobility in Collaboration. In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, pages 305–314, New York, NY, USA, 1998. ACM. doi:10.1145/289444.289505.
- [78] A. May, V. Mitchell, S. Bowden, and A. Thorpe. Opportunities and Challenges for Location Aware Computing in the Construction Industry. In *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*, MobileHCI '05, pages 255–258, New York, NY, USA, 2005. ACM. doi:10.1145/1085777.1085825.
- [79] N. Melão and M. Pidd. Use of business process simulation: A survey of practitioners. *Journal of the Operational Research Society*, 54(1):2–10, 2003. doi:10.1057/palgrave.jors.2601477.

- [80] S. Mitrović-Minić. *The dynamic pickup and delivery problem with time windows*. PhD thesis, Simon Fraser University, Burnaby, BC, Canada, 2001.
- [81] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997. doi:10.1016/S0305-0548(97)00031-2.
- [82] R. Montemanni, L.M. Gambardella, A.E. Rizzoli, and A.V. Donati. A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System. In *Second International Workshop on Freight Transportation and Logistics (ODYSSEUS'03)*, volume 1, pages 27–30, 2003. Available from: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.5826>.
- [83] M. Netjes, W.M.P. van der Aalst, and H.A. Reijers. Analysis of resource-constrained processes with Colored Petri Nets. In *Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 251–265, 2005.
- [84] S. Neudert. Konzeption und Entwicklung eines grafischen Editor-Plug-Ins für mobile Prozesse für die Eclipse Plattform. Master's thesis, Universität Leipzig, December 2010.
- [85] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, USA, 1992.
- [86] I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. PhD thesis, Northwestern University, Evanston, IL, USA, 1976.
- [87] G. Pankratz. Dynamic Planning of Pickup and Delivery Operations by means of Genetic Algorithms. Technical report, Fernuniversität Hagen, 2004. Available from: <http://deposit.fernuni-hagen.de/126/>.
- [88] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008. doi:10.1007/s11301-008-0033-7.
- [89] S.N. Parragh, K.F. Doerner, and R.F. Hartl. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58(2):81–117, 2008. doi:10.1007/s11301-008-0036-4.

- [90] M. Pesic and W.M.P. van der Aalst. Towards a Reference Model for Work Distribution in Workflow Management Systems. In E. Kindler and M. Nüttgens, editors, *Proceedings of the Workshop on Business Process Reference Models (BPRM2005)*, pages 30–44, 2005.
- [91] M. Pesic and W.M.P. van der Aalst. Modelling work distribution mechanisms using Colored Petri Nets. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(3):327–352, 2007. doi:10.1007/s10009-007-0036-z.
- [92] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Universität Darmstadt, 1962.
- [93] M. Polacek, R.F. Hartl, K.F. Doerner, and M. Reimann. A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of Heuristics*, 10(6):613–627, 2004. doi:10.1007/s10732-005-5432-5.
- [94] J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993. doi:10.1016/0377-2217(93)90221-8.
- [95] A.C. Regan. *Real-time information for improved efficiency of commercial vehicle operations*. PhD thesis, University of Texas, Austin, TX, USA, 1997.
- [96] H.A. Reijers, J.H.M. Rigter, and W.M.P. van der Aalst. The Case Handling Case. *International Journal of Cooperative Information Systems*, 12(3):365–392, 2003.
- [97] T. Richter. Regionaler Netzbetrieb der Zukunft. Technical report, Internal document, ENSO - Energie Sachsen Ost AG, Dresden, Germany, February 2008.
- [98] C. Richter-von Hagen and W. Stucky. *Business-Process- und Workflow-Management: Prozessverbesserung durch Prozess-Management*. Vieweg+Teubner Verlag, 2004.
- [99] S. Ropke and D. Pisinger. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40(4):455–472, 2006. doi:10.1287/trsc.1050.0135.
- [100] N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcão e Cunha, editors, *Advanced*

- Information Systems Engineering*, volume 3520 of *Lecture Notes in Computer Science*, pages 11–42. Springer Berlin / Heidelberg, 2005. doi:10.1007/11431855\_16.
- [101] M.W.P. Savelsbergh. The vehicle routing problem with time windows: minimizing route duration. Technical report, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1991.
- [102] M.W.P. Savelsbergh and M. Sol. The General Pickup and Delivery Problem. *Transportation Science*, 29(1):17–29, 1995. doi:10.1287/trsc.29.1.17.
- [103] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics*, 159(2):139–171, 2000. doi:10.1006/jcph.1999.6413.
- [104] S. Schulz. Konzeption und Implementierung eines Verfahrens zur visuellen Manipulation von Geodaten mittels Eclipse Plugin. Master’s thesis, Universität Leipzig, November 2009.
- [105] H. Schuschel and M. Weske. Fallbehandlung: Ein Neuer Ansatz zur Unterstützung Prozessorientierter Informationssysteme. In J. Desel and M. Weske, editors, *Promise*, volume 21 of *LNI*, pages 52–63. GI, 2002. Available from: <http://subs.emis.de/LNI/Proceedings/Proceedings21/article556.html>.
- [106] P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK, 1997.
- [107] Y. Shen, J.-Y. Potvin, J.-M. Rousseau, and S. Roy. A computer assistant for vehicle dispatching with learning capabilities. *Annals of Operations Research*, 61(1):189–211, 1995. doi:10.1007/BF02098288.
- [108] M.E. Shin, A.H. Levis, and L.W. Wagenhals. Transformation of UML-based System Model to Design/CPN Model for Validating System Behavior. In *Proceedings of the Workshop on Compositional Verification of UML Models at the Sixth International Conference on the Unified Modeling Language (CompUML’03)*, 2003.
- [109] M.M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987. doi:10.1287/opre.35.2.254.

- [110] M.Z. Spivey and W.B. Powell. The Dynamic Assignment Problem. *Transportation Science*, 38(4):399–419, 2004. doi:10.1287/trsc.1030.0073.
- [111] A. Sprecher and A. Drexel. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107(2):431–450, 1998. doi:10.1016/S0377-2217(97)00348-2.
- [112] T.S. Staines. Intuitive Mapping of UML 2 Activity Diagrams into Fundamental Modeling Concept Petri Net Diagrams and Colored Petri Nets. In *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'08)*, pages 191–200. IEEE Computer Society, 2008. doi:10.1109/ECBS.2008.12.
- [113] H. Störrle and J.H. Hausmann. Towards a Formal Semantics of UML 2.0 Activities. In P. Liggesmeyer, K. Pohl, and M. Goedicke, editors, *Software Engineering 2005 (SE'05)*, volume 64 of *LNI*, pages 117–128, Bonn, Germany, 2005. Gesellschaft für Informatik, GI-Edition.
- [114] É. Taillard, L.M. Gambardella, M. Gendreau, and J.-Y. Potvin. Adaptive memory programming: A unified view of metaheuristics. *European Journal of Operational Research*, 135(1):1–16, 2001. doi:10.1016/S0377-2217(00)00268-X.
- [115] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial Mathematics, Philadelphia, PA, USA, 2002.
- [116] K. Tumay. Business process simulation. In *Proceedings of the 28th Winter Simulation Conference, WSC'96*, pages 93–98. IEEE Computer Society, 1996. doi:10.1145/256562.256581.
- [117] C. Ulbricht. Startlösungen für das Mobile Workforce Scheduling Problem with Multitask-Processes. Master's thesis, Hochschule für Technik, Wirtschaft und Kultur Leipzig, January 2010.
- [118] M.F. Ursu, B. Virginas, and C. Voudouris. Distributed Resource Allocation via Local Choices: General Model and a Basic Solution. In M. Negoita, R. Howlett, and L. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3215 of *Lecture Notes in Computer Science*, pages 764–771. Springer Berlin / Heidelberg, 2004. doi:10.1007/978-3-540-30134-9\_102.
- [119] W.M.P. van der Aalst. Petri-net-based Workflow Management Software. In *Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems*, pages 114–118, 1996. Available from: <http://lsdis.cs.uga.edu/activities/NSF-workflow/>.



- [120] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998. doi:10.1142/S0218126698000043.
- [121] W.M.P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computers in Industry*, 39(2):97–111, 1999. doi:10.1016/S0166-3615(99)00007-X.
- [122] W.M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '01, pages 42–51, New York, NY, USA, 2001. ACM. doi:10.1145/500286.500296.
- [123] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005. doi:10.1016/j.is.2004.02.002.
- [124] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. The MIT Press, Cambridge, MA, USA, 2004.
- [125] W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005. doi:10.1016/j.datak.2004.07.003.
- [126] F. Vernadat. UEML: towards a unified enterprise modelling language. *International Journal of Production Research*, 40(17):4309–4321, 2002. doi:10.1080/00207540210159626.
- [127] C. Voudouris, G.K. Owusu, R. Dorne, C. Ladde, and B. Virginas. ARMS: An automated resource management system for British Telecommunications plc. *European Journal of Operational Research*, 171(3):951–961, 2006. doi:10.1016/j.ejor.2005.01.010.
- [128] T. Weilkiens. *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Morgan Kaufmann, Burlington, MA, USA, 2007.
- [129] S.A. White. Introduction to BPMN. *BP Trends*, July 2004.
- [130] T.J. Williams. The Purdue enterprise reference architecture. *Computers in Industry*, 24(2-3):141–158, 1994. doi:10.1016/0166-3615(94)90017-5.
- [131] A.M. Wilson. The role of mystery shopping in the measurement of service performance. *Managing Service Quality*, 8(6):414–420, 1998. doi:10.1108/09604529810235123.

- [132] N.H.M. Wilson. Scheduling algorithms for a dial-a-ride system. Technical report, Urban Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, 1971.
- [133] J. Yang, P. Jaillet, and H.S. Mahmassani. On-Line Algorithms for Truck Fleet Assignment and Scheduling Under Real-Time Information. *Transportation Research Record: Journal of the Transportation Research Board*, 1667(1):107–113, 1999. doi:10.3141/1667-13.
- [134] K.-K. Yang and C.-C. Sum. A comparison of resource allocation and activity scheduling rules in a dynamic multi-project environment. *Journal of Operations Management*, 11(2):207–218, 1993. doi:10.1016/0272-6963(93)90023-i.
- [135] K.-K. Yang and C.-C. Sum. An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multiproject environment. *European Journal of Operational Research*, 103(1):139–154, 1997. doi:10.1016/s0377-2217(96)00266-4.
- [136] M. zur Muehlen. Organizational Management in Workflow Applications - Issues and Perspectives. *Information Technology and Management*, 5(3):271–291, 2004. doi:10.1023/B:ITEM.0000031582.55219.2b.