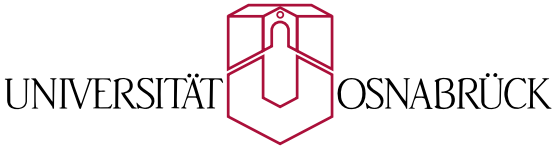

MODELING AND EXECUTION OF RESILIENT BUSINESS PROCESSES IN UNRELIABLE COMMUNICATION ENVIRONMENTS

Frank Nordemann

Dissertation zur Erlangung des Doktorgrades (Dr. rer. nat.)
des Fachbereichs Mathematik/Informatik
der Universität Osnabrück

Einreichung: 8. November 2021
Disputation: 23. Februar 2022

Erste Gutachterin:	Prof. Dr.-Ing. Elke Pulvermüller
Zweiter Gutachter:	Prof. Dr.-Ing. Ralf Tönjes
Dritter Gutachter:	Prof. Dr. Pedro José Marrón

Dedicated to Julia, Marlene, and Hanne.

ACKNOWLEDGMENTS

Many people encouraged and escorted me in the process of writing this thesis. I am very grateful for the help and support of every one of them.
I would like to deeply thank Prof. Dr. Elke Pulvermüller for supervising my work. As my doctoral advisor, she supported me with valuable and productive feedback starting from the first ideas up to the final thesis. I am grateful for the discussions, the ideas, and the trust I received from her along the way. I deeply thank Prof. Dr. Ralf Tönjes for encouraging and supporting me in writing a Ph.D. thesis. Discussing research ideas and problem statements has been an inspiration for finding the solutions presented in this thesis. I would like to extend my thanks to Prof. Dr. Pedro José Marrón for taking the role of the third reviewer.
This thesis has been developed as part of the research projects OPeRAte and OPeRAte-Plus. The insights into the real-world challenges of collaborative agricultural processes helped me in finding appropriate process examples for designing and evaluating solution approaches. I would like to thank Prof. Dr. Heiko Tapken, Franz Kraatz, and Maik Fruhner for motivating and supporting me in writing this thesis.
Being part of the lab for mobile communications at the Osnabrueck University of Applied Sciences is fun. I always enjoyed working collaboratively on exciting research challenges with my colleagues. I want to thank Thorben Iggena, Marten Fischer, Daniel Kümper, Anas bin Muslim, Sarmad Ghafoor, Julian Dreyer, Manuel Kerkmann and Günter Hüdepohl for their support, their feedback and all the fun times off work.
Writing a Ph.D. thesis is a thrilling challenge, including many highs and lows along the way. I am deeply thankful for the unwavering support of my wife Julia. She understands to motivate and inspire me at all times - without her, this work would not have been possible. Thank you so much, Julia.

ABSTRACT

Business processes define workflows by structuring sets of activities according to given objectives. Process Modeling Languages (PMLs) provide graphical elements to define process models. Apart from use cases in finance and commerce, PMLs gain popularity in application domains such as Cyber-Physical Systems, the Internet of Things, ubiquitous computing, mobile devices, and scenarios happening in rural, restricted, or disaster-affected regions. Many of the domains are exposed to delayed, intermittent, or broken connectivity. Existing PMLs show limitations in considering connectivity-related issues, leading to failures and breakdowns at process runtime.

This thesis addresses connectivity-related issues regarding the **modeling and execution of resilient business processes taking place in unreliable communication environments**. With *resilient BPMN* (*rBPMN*), an extension for the Business Process Model and Notation (BPMN) addressing environments with delayed, intermittent, or broken connectivity is introduced. *rBPMN* extends the BPMN metamodel by new elements for resilient process models. Domain experts may define alternatives for possibly failing message flows based on priorities or characteristics of the alternatives. Functionality offered by remote participants may be moved to other participants for local execution. This thesis illustrates approaches for the graph-based analysis of business processes regarding their resilient operation. By translating process models into directed graphs, graph algorithms allow to dynamically find the most suitable process path. Domain experts are enabled to identify non-resilient parts of a process model, allowing them to optimize the involved segments before runtime. Multi-criteria analysis approaches optimize process operation according to a chosen set of criteria.

A real-world scenario of an environmental-friendly slurry application illustrates the use of *rBPMN's* concepts and approaches for modeling and executing resilient processes. Technical approaches realizing *rBPMN's* resilience strategies using a BPMN runtime engine and microservices are illustrated. The proof-of-concept implementations may be extended and adapted, serving as guides for other application domains.

CONTENTS

CHAPTER 1

INTRODUCTION

This chapter motivates the need for the modeling and execution of resilient processes in unreliable communication environments. The effects of unreliable connectivity on collaborative business processes are illustrated, followed by the identification of challenges to be addressed. Further on, the research scope of this thesis and its structure are outlined.

1.1 Motivation

A business process combines and structures a set of activities into an orchestrated workflow. Such workflows assist in achieving defined objectives, such as manufacturing goods, selling products, and processing data. The definition, execution, monitoring, optimization, and documentation of business processes is a well-known field of activity in many companies and organizations. Referred to as Business Process Management (BPM), it facilitates the realization of corporate goals and visions. Business processes often include different participants, such as the companies' customers and business partners. They enable the organization of collaborative work and allow the adaptation of process operation based on metrics, often defined as Key Performance Indicators (KPIs). A common practice for the definition of business processes is to apply a *Process Modeling Language (PML)* [168]. The modeling elements offered by the chosen PML allow the creation of a process model, defining the activities and structure of the process

workflow. Operation and control of the model are supported by process runtime engines, available for different PMLs.

Some application domains have a long tradition in using and refining business processes. In the finance industry, the creation of bank accounts and the realization of cash withdrawals, bank transfers, and credit applications are typical business processes. In the area of selling goods, business processes for the implementation of orders, inventory management, payments, and shipping are widely used. Other traditional application domains of business processes are the processing of returned goods (verification, payment, inventory), booking of tickets/services (e.g., flights, concerts), quality assurance, and the acquisition of clients.

The *Business Process Model and Notation (BPMN)* [125] is a major PML widely used in academia and industry [5] [171] [180]. With its well-defined set of modeling elements, its expressiveness and usability for domain experts, its non-proprietary and extensible design, the mature tool support and ISO-Certification [91], it is increasingly being applied also for business processes outside the traditional domains of finance and commerce. Examples include Cyber-Physical Systems (CPS) [82], the Internet of Things (IoT) [34], Wireless Sensor Networks (WSN) [28] [41] [160], ubiquitous computing [176] [177] [179], disaster relief management [11], manufacturing of goods [1], healthcare and mobile devices [19] [180]. A substantial part of organizing collaborative work is the exchange of messages and data among participants. However, some of these application domains experience connectivity issues when communicating with other participants. Rural surroundings, isolated areas, and limited device capabilities often result in executing business processes in unreliable communication environments, where connectivity is limited, intermittent, delayed, or broken.

Unreliable communication is a major risk for the operation of business processes. Even short delays or timely limited interruptions of connectivity can lead to unexpected behavior in process operation. Many processes that are not designed for unreliable communication environments quickly experience complete breakdowns of operation. Depending on the process scenario, typical consequences are cost-intensive downtimes of machinery, on-hold workers, the loss of goods, and contractual penalties for the late completion of the process or the delivery of goods and services.

Communication-resilient design and execution of business processes are required to prevent process failures based on connectivity issues. However, existing PMLs such as BPMN are not designed for the challenges of unreliable communication in particular. While BPMN allows some flexibility to address connectivity issues, modeling is often cumbersome, requires expert-level modeling skills, is restricted in its capability to consider dynamics, and is often bound to a specific scenario and its characteristics. For instance, a process model may operate well in a given scenario, but insufficiently in

another because of different connectivity characteristics of the two scenario surroundings. Another process model may fail during operation due to its missing capability to dynamically replace a missing participant with an available equivalent. Also, a disadvantage of BPMN is the absence of a mechanism to verify communication-resilient operation at design time and hence to optimize resilience before process execution. Imperfections are identified during execution when running processes interrupt or break down.

A PML addressing the challenges of unreliable communication environments could assist domain experts in modeling communication-resilient process models, preventing process failures at runtime and their cost-intensive consequences. Dynamic adaptations of process operation could not only enhance resilience, but additional aspects of operation (e.g., accuracy, cost, time). Depending on the individual scenario, this may lead to faster process operations, a minimized usage of resources, and an increased product quality.

The literature describes many different PMLs for the definition of business processes [115]. Due to its representation of a comprehensive PML and its widespread usage in academia and industry, this thesis uses BPMN as PML [5] [171] [180].

1.2 Problem Statement

Modeling and execution of collaborative business processes operating in unreliable communication environments are challenging. Delayed, intermittent, or failing communication with other participants possibly leads to process interruptions or complete breakdowns. Different challenges exist for the modeling and execution of resilient processes. Subsequently, the challenges are grouped into four categories:

Challenge 1: Modeling of Resilient Processes

> Unreliable communication is not a focus of BPMN. A resilient process model needs to cope with potentially interrupted and unavailable communication with other participants. However, modeling elements ensuring resilient operation in the case of delayed, intermittent, or broken connectivity are missing and have to be developed. Further on, modeling elements capable of adapting a process for optimal operation under the given circumstances are missing. Such modeling elements need to address scenario-specific characteristics such as dynamic process behavior, allowing the reuse of the same process model in multiple scenarios. Optimal process operation may also require respecting additional criteria such as accuracy, cost, and time along with resilience. New modeling elements have to be carefully designed and integrated into the existing BPMN modeling palette.

Domain experts should not lose their focus on the design of the actual workflow due to the complex use of resilience-oriented modeling elements.

Challenge 2: Process Resilience Verification

A major aspect in avoiding process failures based on connectivity issues is to measure and verify the communication resilience of a process model. By verifying the model before process runtime, domain experts can adjust and improve the model. However, a verification mechanism and corresponding metrics to verify resilient operation are missing. A verification mechanism needs to consider varying connectivity characteristics of different scenarios. It is unknown how appropriate metrics to measure and compare/rank the resilience of different process paths may look like. At process runtime, a mechanism which identifies upcoming non-resilient process segments and adapts operation accordingly is missing.

Challenge 3: Multi-Criteria Process Operation

While resilience is an essential aspect of processes taking place in unreliable communication environments, it is not the only criterion of interest for process operation in most scenarios. An optimal operation of a process may consider multiple criteria, also known as KPIs. The challenge is to create/identify/combine appropriate metrics and analysis methods to evaluate and compare different process paths regarding a set of chosen criteria. A decision-making approach for selecting the best-suited process path from several eligible paths fulfilling the requirements of the different criteria is missing.

Challenge 4: Resilient Process Execution

The process model defines elements that can be used to ensure resilience in unreliable communication environments. However, it is not defined what technologies and architectural principles are suitable for their realization. The BPMN metamodel includes no engineering specifications describing how to implement the modeling elements. Besides, the implementation of supporting tools and interfaces in non-application layers may be required for resilient process execution. For instance, a running process needs to be aware of available and dynamically appearing/disappearing neighboring participants. This information is part of the network layer of the ISO/OSI model, requiring an interface to the process execution environment located in the application layer.

1.3 Research Focus

The *main objective* of this thesis is to *provide concepts and approaches to avoid interruptions, failures, and breakdowns of business processes caused by the effects of unreliable connectivity*. Based on the problem statement and the challenges raised in section 1.2, strategies for the modeling and execution of resilient processes in unreliable communication environments are elaborated and evaluated. While the developed approaches introduced in this thesis may also work in other PMLs, this thesis focuses on BPMN as PML.

The main objective of this thesis summarizes several sub-objectives, namely:

- Modeling concepts for the extension of BPMN, addressing the challenges of modeling resilient processes (*Challenge 1*);
- A resilience verification approach for process models, including resilience metrics and analysis methods (*Challenge 2*);
- An analysis approach identifying the process path of optimal operation, considering resilience along with other criteria of process operation like accuracy, cost, and time (*Challenge 3*);
- A proof-of-concept, evaluating the developed concepts and approaches in a real-world scenario (*Challenge 4*);
- Recommendations for the modeling, analysis, and execution of resilient business processes (part of *Challenges 1-4*).

1.4 Thesis Outline

The remainder of this thesis is organized as follows: **Chapter 2** (p. 9ff.) provides the fundamentals and related work relevant to this thesis. The characteristics of unreliable communication environments are illustrated, along with an agricultural real-world example of an environmental-friendly slurry application process. The fundamentals of BPMN are presented. BPMN extensions published in the literature are identified and examined regarding their suitability for unreliable communication environments. A detailed examination of BPMN and its limitations for unreliable communication environments is illustrated by modeling and evaluating the slurry application process in BPMN. The chapter concludes by presenting fundamentals and related work regarding the analysis of business processes and their execution.

After the fundamentals and related work in chapter 2, the following four chapters present the main contributions of this thesis and address the challenges identified for

the modeling and execution of resilient business processes. The relations of the different chapters to each other and to the challenges are depicted in Figure 1.1.

Chapter 3 (p. 41ff.) introduces *resilient BPMN* (*rBPMN*), a BPMN extension for the modeling of resilient business processes in unreliable communication environments. A set of metamodel concepts and their graphical representations allow domain experts to ensure resilience by handling possibly breaking message flows and adapting process operation accordingly. The modeling elements allow experts to decide statically or dynamically on the optimal process operation.

A graph-based approach for the resilience analysis of business processes is the subject of **chapter 4** (p. 65ff.). At first, definitions of communication-resilient processes and appropriate metrics are provided. Following, a rule set for the translation of process models to graphs is introduced. The chapter concludes by illustrating the resilience analysis using different graph-based search algorithms.

The resilience analysis is expanded to consider additional criteria along with resilience in a graph-based analysis in **chapter 5** (p. 95ff.). Multi-criteria metrics to measure and compare different process paths are introduced. The process-to-graph translation rules are adapted for multi-criteria aspects. Following, different approaches for the multi-criteria analysis are illustrated.

Chapter 6 (p. 123ff.) focuses on the resilient execution of process models. After identification of the requirements, resilience strategies for process execution are introduced. The strategies consider aspects not realized by the process model. This includes the initial participant configuration, movement and usage of functionality across participants, and the dynamic discovery of neighboring participants. Besides, the chapter covers the optimization of processes at runtime using graph-based and non-graph-based decision-making methods.

In general, business processes are analyzed and optimized using a variety of techniques and approaches. Due to the potential chapter 2 identifies in this technique, this thesis focuses on graph-based solutions for process analysis and decision-making (addressing challenges 2 and 3). However, the approaches for modeling and executing resilient processes are not limited to graph-based decision-making. Throughout this thesis, an alternative technique is applied for decision-making in business processes.

Evaluation of the modeling concepts and approaches for analysis and execution is part of **chapter 7** (p. 139ff.). The environmental-friendly slurry application introduced in chapter 2 is optimized for resilient and multi-criteria optimal operation by using *rBPMN* modeling elements. Following, a proof-of-concept illustrates the implementation of *rBPMN's* resilience strategies using the BPMN runtime environment Camunda [25] and the architectural principle of microservices. Performance and scalability of the graph-based analysis approaches are evaluated using a process graph

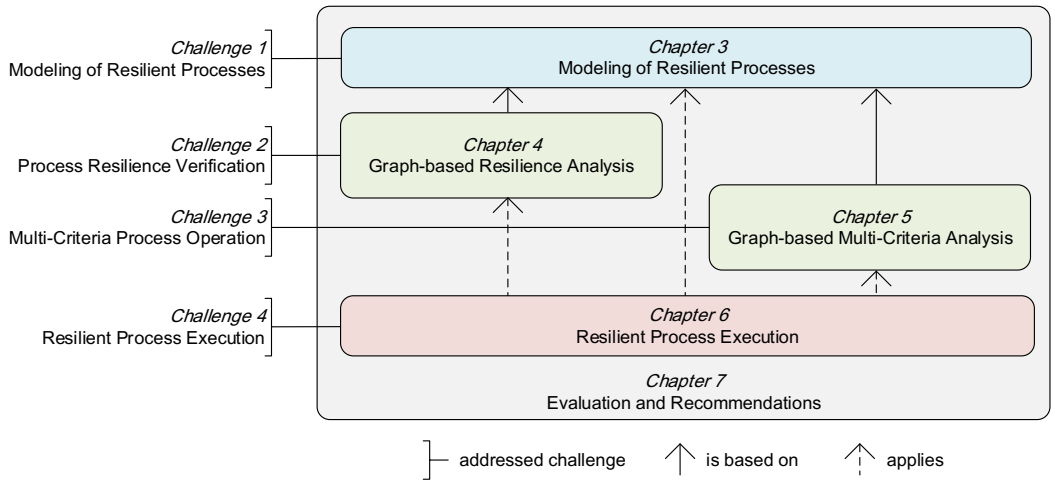

Figure 1.1: Relations between the main technical chapters of this thesis and the addressed challenges.

generator and different graph-based search algorithms. Finally, recommendations for the use of *rBPMN* and its resilience strategies are given.

Chapter 8 (p. 191ff.) concludes this thesis by summarizing the challenges, contributions, and findings. Concepts and approaches are briefly discussed before directions of future work are presented.

Figure 1.1 illustrates the relations between the main technical chapters and to the identified challenges. The decision-making approaches for resilience and multi-criteria optimization introduced in chapters 4 and 5 are based on the process modeling concepts presented in chapter 3. A process model includes information regarding estimated and required connectivity, which is processed by the resilience analysis. Further on, a model defines the relevance of different criteria for the modeling elements and guides the multi-criteria analysis accordingly. Chapter 6 covers the execution of business processes. Hence, it integrates approaches of the graph-based analyses (chapters 4 and 5) and the actual process model itself, presented in chapter 3. Finally, concepts and approaches of the chapters 3 to 6 are combined and investigated in the evaluation chapter 7.

CHAPTER 2

FUNDAMENTALS AND RELATED WORK

This chapter presents the fundamentals and related work relevant to this thesis. First of all, unreliable communication environments and their characteristics are presented. Examples of communication technologies applicable for unreliable environments are given. The issue of finding a communication path between participants in an unreliable environment is outlined. With an environmental-friendly slurry application, a real-world collaborative process taking place in an unreliable communication environment is presented.

The remainder of the chapter illustrates fundamentals and related work concerning the four challenges of this thesis, identified in section 1.2 (p. 3f.). The objective is to briefly present the state of the art and to illustrate open research issues. Paragraphs of *italic text* conclude open research issues and separate the work presented in this thesis from the state of the art.

Challenge 1 is addressed by illustrating PMLs in general and BPMN in particular. By presenting extensions available for the BPMN metamodel, related work for the domain-specific adaptation of BPMN is described. Modeling of a slurry application process outlines limitations of BPMN regarding unreliable connectivity. Afterward, the following section relates to challenges 2 and 3, namely verifying resilient and multi-criteria process operation. Algorithms applicable for analyzing graph data structures are presented and related work considering graphs and BPMN is discussed. Finally, the execution of business process models using tools for BPM is presented, addressing challenge 4. Open issues in the resilient execution of BPMN processes are identified.

2.1 Unreliable Communication Environments

In unreliable communication environments, communication between process participants is prone to delays, interruptions, and failures. Connectivity may be lost dynamically or be completely absent. This is a challenge for many business processes. While in cloud environments services may be replaced quickly by backup services, this may not be possible in unreliable environments due to the lack of other service-offering participants.

This section illustrates application domains of unreliable communication environments. Characteristics of and technologies used for communication are presented, followed by insights on pathfinding in dynamically changing networks. The section concludes by presenting a collaborative agricultural process as an example of a process taking place in an unreliable environment.

2.1.1 Characteristics and Application Domains

The label *unreliable communication environment* is used as a collective term for various scenarios experiencing connectivity issues in this thesis. Different kinds of communication networks are applied in such scenarios, all exposed to the effects of unreliable communication. Examples for communication networks include Mobile Ad-hoc Networks (MANETs) [7], Delay Tolerant Networks (DTNs) [66] [68], Opportunistic Networks (OppNets) [88] [162], hybrid networks [90] [111] and Wireless Sensor Networks (WSNs) [174].

The major aspect of unreliable communication environments is the *limited, delayed, intermittent*, or *broken* connectivity between network nodes [88] [66] [68] [90]. Connectivity issues are often directly related to characteristics of the network nodes, such as their mobility behavior and technical configuration. For instance, nodes may be limited in their communication capabilities due to performance restrictions or energy-consumption constraints [174].

Besides, the characteristics of the environment may have significant effects on communication. Many scenarios take place in rural, remote, or sparsely populated locations, where infrastructure-based connectivity lacks coverage [83] [105] [130] [94]. Also, environments may impact the quality and strength of communication signals. Scenarios located in mountain regions, below ground, underwater, in woodlands, or inside buildings experience signal distortions when communicating wirelessly [3]. Node mobility may further reduce the number of communication possibilities. Especially in scenarios where nodes meet dynamically for short periods, communication is challenged. This leads to constantly changing network topologies.

In terms of business processes, network nodes represent participants of a collaborative workflow. While participants may be human or non-human, they often represent systems of the same or different organizations.

Scenarios taking place in unreliable communication environments originate from different kinds of application domains [3]. Typical examples include scenarios in rural areas like agriculture [130], forestry [96], mining [78], wildlife and environmental monitoring [161] as well as scenarios including limited devices such as IoT and CPS. Logistical processes located in remote locations may be affected. Additional examples are scenarios taking place in disaster-affected or politically-restricted environments, where communication infrastructure may fail dynamically on a large scope. Scenarios often indicate connectivity as opportunistic, describing its unplanned and random characteristics [88] [162].

While the application domains and use cases are diverse, scenarios taking place in unreliable communication environments show certain similarities. Although it is not a requirement, most scenarios include wireless technologies for communication. Wireless network segments are often combined with wired nodes, such as nodes located in the internet. However, similar connectivity issues may be identified in scenarios where wired segments of a network get dynamically connected and disconnected. In unreliable communication environments, nodes have to be capable of managing delayed, intermittent, or broken connectivity with other nodes. Depending on the scenario, connectivity failures may be limited to specific periods/locations or may happen at any time. Due to the constant risk of connectivity failures, the implementation of mechanisms for resuming data transfers is advised. As a result, the need for re-transmissions can be reduced.

2.1.2 Technologies

A variety of communication technologies may be used in unreliable communication environments. Many scenarios combine infrastructure-based (e.g., cellular, WiFi in access-point mode) and infrastructure-free (ad-hoc) communication technologies, forming hybrid networks based on the nodes' connectivity capabilities and mobility characteristics [58] [90] [111]. Properties such as available bandwidth, latency, and packet loss may differ between the used technologies.

Cellular communication is often used connecting mobile participants to other participants located in the cloud. Ad-hoc communication in the form of MANETs allows participants to communicate dynamically with each other without the presence of a communication-enabling infrastructure component. MANETs often enhance opportunities for communication, especially in highly dynamic scenarios.

Unreliable communication between nodes often results in fragmented networks, where some segments of the network miss constant connectivity to other segments. End-to-end connections between nodes of the network are missing, resulting in communication failures for the data transfers to the corresponding nodes [68] [90]. Originating from the area of space networks, the *Bundle protocol* [23] enables participants to communicate with each other even in the case of missing a continuous communication path between them. In space, communication requires a direct line-of-sight. Due to rotations of planets and communicating infrastructure such as satellites and rovers, a direct communication path is often missing. Hence, data packets are transported to intermediate hops using the *store-carry-forward / custody principle* [67], until the packets reach the destination node. For instance, a rover may send its data to a satellite, which directs the packets to satellite dishes located on earth. While there may not be a contentious path at any time, communication is realized at the cost of high delay times.

The Bundle protocol inserts a Bundle layer on every node of the network. Inserted between the transport and the application layer of the ISO/OSI model, it manages the data transfers between nodes. Application data is packed into so-called bundles when reaching the bundle layer. By adding control information and a Bundle protocol header, the destination-addressing information is accessible for every node on the path to the destination. The Bundle protocol uses the lower layers to transfer the data to the next hop. Every intermediate hop may offer a store-carry-forward database, temporarily storing packets on their way to the destination node.

The operating principle of the Bundle protocol quickly results in high delay times between source and destination. Applications have to be designed for such high delay times. Interactivity between nodes needs to be reduced or eliminated, asynchronous communication is advised.

2.1.3 Pathfinding

A challenge in unreliable communication environments is to find a communication path between the source and destination of a data transfer. Depending on the network technologies employed in the environment, different routing algorithms are available and need to be configured, adapted, and combined [133] [142] [92] [104]:

MANET Routing Algorithms

In MANETs, the identification of neighbor nodes is essential for communication. Different routing approaches specialized for MANETs exist. With proactive routing algorithms like *Destination-Sequenced Distance-Vector (DSDV)* [137] and *Optimized Link State Routing (OLSR)* [38], routing information is exchanged regularly between nodes.

So-called *hello messages* are broadcasted using a configurable frequency, allowing other nodes within the communication coverage range to pick up the broadcasted information and hence the existence of a neighboring node. The message is processed to add the neighbor node into a local neighbor table, collecting the addressing information of all currently available neighbor nodes. Depending on the routing algorithm, additional information about intermediate hops to other destinations in the MANET may be enclosed.

With reactive routing algorithms like *Dynamic Source Routing (DSR)* [97] and *Ad-hoc On-demand Distance Vector (AODV)* [136], no regular broadcast messages exist. Topology information is not collected proactively. Instead, paths to the destination are identified at the time needed by emitting route requests to a given destination. While this principle avoids overhead for building up neighbor tables which may not be needed, it prevents the gathering of topology information when there is no need for communication.

Further approaches for MANET routing exist, often closely related to the present scenario. For instance, data may be forwarded according to geographical locations [100] or based on social aspects [118].

Delay-Tolerant Routing Algorithms

Due to the absence of a continuous communication path, data delivery is usually not guaranteed in DTNs. Stochastic routing algorithms aim at increasing the probability of a successful data transfer [93] [181]. With epidemic routing [165], data packets are copied to every node met in the network. Eventually, a node storing a data packet copy meets the destination node and delivers the data. Since packets are copied on a large scale using epidemic routing, strategies to limit the creation of packet copies and to remove old copies from the network are required. Otherwise, the network may quickly overload and become dysfunctional. A variant that allows controlling the number of packets is represented by Spray & Wait routing [157]. With PRoPHet routing [101], the recent history of node encounters is analyzed to predict future encounters.

Apart from stochastic routing, other approaches for routing in DTNs exist. Geographical routing approaches integrate locations to navigate data packets. Social-based approaches integrate social aspects into routing [89] [182]. Biological-inspired algorithms exploit concepts such as swarm intelligence for routing [12].

Hybrid Routing

A cellular network often connects mobile nodes with other nodes located in cloud environments. Also, cellular communication technologies may be used to connect multiple

mobile nodes among each other. However, in hybrid networks, a cellular network may be combined with the local MANET present at a node [111]. To take advantage of possible communication opportunities between a local MANET and nodes present in the cloud, the node connecting to the cellular network and the MANET needs to gateway requests across both networks. To realize communication, routing algorithms of both networks need to respect nodes of the other network.

2.1.4 An Environmental-Friendly Slurry Application

An agricultural scenario of an environmental-friendly slurry application is used for illustrating a collaborative process taking place in an unreliable communication environment. Based on regulations of the European Union, legal guidelines have to be addressed when applying slurry to fields in Europe. The objective is to prevent over-fertilization and its negative impact on the environment.

Regulations limit the amount of fertilizer that can be applied to a field. In many regions, the amount of nitrogen as part of the slurry is limiting its applicable amount on a field. In other regions, phosphorus is the limiting factor. Since slurry is a mixture of different ingredients, an *ingredients analysis* needs to identify its chemical structure. Afterward, the slurry amount containing the allowed quantities of nitrogen and phosphorus can be calculated. The chemical structure of the slurry depends on the type of animal it is originating from. Further on, storage and processing of the slurry have a big impact. For instance, even in the same barrel, differences between slurry located at the top and at the bottom exist if it has not been stirred recently. Most accurate analysis results are gained by analyzing the slurry in a laboratory. However, this is also a costly and time-consuming procedure. Alternatively, a near-infrared spectroscopy sensor may provide solid analysis results on the fly. Another approach is a manually performed rapid test, delivering results within minutes. A less accurate approach is to look up the chemical structure in ingredients reference tables. Improved results may be provided by integrating additional information about the animals' feed.

In the past, the same amount of slurry was applied to all sections of a field. Due to varying compositions of the soil, plants are able to process the slurry in some sections better than in other sections. For instance, on sandy sections of the field, the slurry is quickly seeping into the deeper ground and into the groundwater. In areas with clay soil, plants have more time to access the slurry's ingredients. With geographical data about the soil and the yields of past harvests available, a precision farming expert is able to create *Application Maps (AppMaps)* for the slurry application. As illustrated in Figure 2.1, AppMaps specify subsections/partfields on a field and add a needs-based amount of fertilizer. This results in an environmental-friendly fertilizing process with a better growth of plants and less slurry seeping into groundwater.

Figure 2.1: An AppMap specifying the applicable amount of nitrogen, being part of a terminal application located on a slurry spreader (courtesy of ANEDO [6]).

To accurately place the correct amounts of slurry onto the intended partfields, a position sensing system is required. While GPS is the standard technology used for positioning the slurry spreader onto the field, its accuracy may be improved by using Real-Time Kinematic (RTK). This requires the reception of a correcting signal, which may be emitted by a station positioned in the proximity of the field. Alternatively, a correcting signal may be received by cellular communication.

An environmental-friendly slurry application process on the slurry spreader needs to communicate to at least three other participants for the *calculation of AppMaps*, the *slurry ingredients analysis*, and the *position sensing and correction*.

Since a fully loaded slurry spreader is often too heavy to drive on public streets, trucks may transport the slurry to the field. Containers at the edge of the field temporarily store the slurry until the slurry spreader is reloading. Deepening on the distance between slurry storage and the fields, a varying number of trucks needs to be organized. A supply chain of slurry delivering trucks supporting the slurry spreader emerges. Large-scale slurry applications employ a central management participant controlling slurry supply chains. Communication with every entity of the slurry process is required to gather status information and to instruct the participants.

Naturally, the slurry spreader is located on the field during application. The same applies to a near-infrared spectroscopy sensor for the slurry analysis and the local position correction station. The AppMap-service offered by a precision farming expert, the reference-based slurry analysis as well as results from a laboratory analysis are accessible in the cloud. A correcting signal may be received by cellular communication. The central management participant is also located in the cloud while supporting trucks are constantly moving between the slurry storage location and the field.

Although this thesis focuses on a process on a slurry spreader including participants for the AppMap creation, the slurry analysis, and the position sensing system, a resilient process execution remains challenging. Located in rural and sparsely populated regions, the availability of cellular communication is not guaranteed on agricultural fields. Connectivity to all participants may be unreliable, depending on the scenario.

2.2 Modeling of Business Processes

This section illustrates the modeling of business processes using PMLs. The fundamentals of BPMN, its integrated extension mechanism, and related work regarding the domain-specific extension of BPMN are presented. Limitations of BPMN regarding unreliable environments are illustrated by modeling an agricultural slurry application process. The section concludes with a summary of findings and remarks regarding open research issues in resilient process modeling.

2.2.1 Process Modeling Languages

A business process can be defined as an orchestration of several different operations [172]. An orchestration may include various ways to traverse from start to end, also called process paths. A path is determined by process variables, getting evaluated at decision points part of the orchestration and defining the process configuration [171]. PMLs may be used for representing a workflow in a standardized notation, defining the elements available for modeling workflows [168]. Examples include traditional flow charts, Petri Nets [103], UML Activity diagrams [128], and Event-driven Process Chains (EPCs) from ARIS [60]. One of the most widely applied PMLs is the BPMN [125].

Many PMLs create business processes by the orchestration of graphical elements in a workflow [171]. The graphical models are often annotated with additional information, describing context and execution instructions. Some PMLs such as BPMN link process activities to source code, allowing to implement business logic used for executing process models.

Usually, processes are controlled and evaluated using different criteria. Criteria are process characteristics such as accuracy, error ratio, volume of data, self-sufficiency

level, performance requirements, resilience, calculation time, failure probability, and monetary cost. Especially in the field of business administration, sets of crucial process criteria are also known as *Key Performance Indicators (KPIs)*. Most process evaluations use a diverse set of criteria, often defined in a prioritized order. The modeling elements of a PML enable the integration of decision points, where the criteria are used to control the flow of a process.

2.2.2 Business Process Model and Notation

BPMN represents a popular PML, capable of being customized for the requirements of different application domains ([125] p. 42). Due to its flexibility, expressiveness and usability for domain experts, its non-proprietary and extensible design, mature tool support and ISO-Certification, BPMN fits many different use cases of business processes. Besides traditional process modeling for banks, logistics, and sales, it is being applied to use cases taking place in unreliable communication environments. Examples include scenarios in rural areas like agriculture [121] and disaster relief management [11] as well as scenarios including limited devices such as CPS [82], IoT [34], and WSNs [28] [41] [160]. Since handling limited, intermittent, or failing connectivity is not in the focus of BPMN, modeling resilient processes is challenging for domain experts of many application domains. Subsequently, the fundamentals of modeling processes using BPMN and applying its integrated extension mechanism are presented.

BPMN Process Modeling

The focus of BPMN is on the graphical definition of the process structure, including activities, gateways, sequence flows, and message flows ([125] p. 25ff.). Domain experts may compose processes including human-based activities and (automated) machine-based activities. In the *BPMN metamodel*, the concept *activity* represents a base concept for many specializations such as script tasks, service tasks, manual tasks, subprocesses, and generic, unspecified tasks ([125] p. 155ff.). When referred to in this thesis, the term activity summarizes all different specializations.

Figure 2.2 illustrates the collaborative process *Slurry Check* (abbreviated: *SC*), modeled in BPMN 2.0.2. The process uses a weather provider to determine the conditions for a slurry application on a field. The italic and red-colored text designates the BPMN elements used in the process model. The elements are explained subsequently.

The process *SC* is placed within a so-called pool, illustrating the system borders of the participant which executes *SC*. The start- and endpoints are represented by circles, the sequence flows in the form of arrows connect the start- and endpoints with two tasks and a decision point. The first task *Check conditions* calls a service at a weather

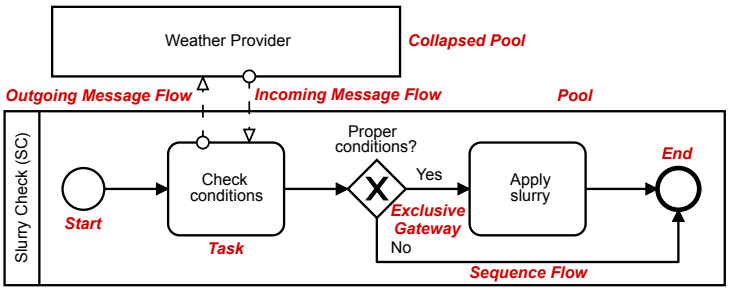

Figure 2.2: Check Slurry (*SC*) process (modeled in BPMN).

provider to query the current weather conditions. An outgoing message flow illustrates the request for weather information, an incoming message flow contains the result sent back by the weather provider. After checking the conditions, a decision point in the form of an exclusive gateway (XOR) determines whether or not the task *Apply slurry* is executed or if the process directly ends. Besides the elements presented here, the BPMN modeling palette includes many more modeling elements ([125] p. 25ff.).

The scenario in Figure 2.2 contains two participants. While the first participant executing *SC* shows all details of the enclosed process, the weather provider is depicted as a collapsed pool. The details of the process executed by the weather provider remain unknown, it may be seen as a black box. While collapsed pools may be used as a tool for abstraction, they also allow modeling collaborative processes of different organizations. No organization has to reveal its process structures and included trade secrets. Further on, technologies used to execute the processes are not defined by BPMN and may vary between participants. For instance, the weather provider may implement the weather service without the use of a BPMN process.

BPMN defines different types of process models, which are also called diagrams. The most widely used type is the *Business Process Diagram (BPD)*, which consists of orchestrating start- and endpoints, decision points, and activities into a workflow. The scenario depicted in Figure 2.2 represents a *collaboration diagram*. Collaboration diagrams are specific BPDs including different participants, communicating with each other using message flows ([125] p. 107ff.). Since connectivity issues involve communication between multiple participants, collaboration diagrams are of major interest in this thesis.

Two other diagram types exist, which are less popular. *Choreography diagrams* can be seen as a type of business contract between participants ([125] p. 315ff.). They do not focus on the orchestration of activities into workflows of different participants, but on interactions between participants. *Conversation diagrams* group message flows that correlate with each other ([125] p. 123ff.). For instance, the message flows taking care

of orders and shipments may be grouped into a conversation. This way, communication between participants may be abstracted from individual message flows as an additional point of view on collaborating processes.

BPMN can be used in conjunction with the *Case Management Model and Notation (CMMN)* [126] and the *Decision Model and Notation (DMN)* [129]. Instead of having a defined workflow from start to end as with BPMN, CMMN allows the modeling of cases that are being controlled by a process user. The notation is an appropriate choice for scenarios in which an automated selection of cases and their ordering is challenging. Further on, DMN allows to model decision-making in the form of decision tables. Process variables may be used as parameters to determine a decision from a list of possibilities.

BPMN is designed following a model-driven approach. Its metamodel defines the concepts, relationships, and semantics of objects (or modeling elements) and is based on the *Meta Object Facility (MOF)* [127] of the Object Management Group (OMG). In addition to the metamodel, there is a second type of process representation. Using the *XML Metadata Interchange (XMI)* standard, process models may be exchanged between different software tools. A detailed description of the concepts, their relationships and semantics of the metamodel as well as the corresponding XML schemas can be found in the BPMN specification document [125].

BPMN was initially released by the OMG in 2006 [123]. In 2011, it has been expanded for additional modeling elements, an *integrated extension mechanism* and execution semantics in version BPMN 2.0 [124]. The latest version BPMN 2.0.2 [125] was released in 2014, containing only minor corrections of BPMN 2.0. All BPMN process models used in this thesis are based on version 2.0.2.

BPMN Extension Mechanism

Since BPMN is designed as a universally applicable PML, some application domains may miss specific modeling elements required to represent characteristics of their scenarios. *BPMN artifacts* can be used to add descriptive information to process models, e.g., using *TextAnnotations* ([125] p. 123ff.). In contrast to artifacts, the integrated BPMN extension mechanism allows to extend the language with new modeling elements ([125] p. 55ff.). While BPMN may be easily extended by modifying its metamodel, using the integrated extension mechanism guarantees the validity of core BPMN elements. This ensures that existing BPMN tools are capable of processing extended models. However, the additional elements and attributes of the extension may be lost in the tools.

As defined by the metamodel for the extension mechanism ([125] p. 55), an extension consists of four different elements:

- *Extension*: Can be seen as a container, binding/importing the extension with its attributes to a BPMN model definition.
- *ExtensionDefinition*: A defined and named group of *ExtensionAttributeDefinitions* to be added to existing BPMN elements.
- *ExtensionAttributeDefinition*: Defines a name for an extension attribute.
- *ExtensionAttributeValue*: Defines the concrete attribute type/value of an *ExtensionAttributeDefinition*. May reference existing BPMN types or data types.

The BPMN specification is missing *methodological guidelines* on how to use the extension mechanism. Additional challenges such as inconsistencies between the extension definitions in the metamodel and the XML Schema Definition (XSD) exist (cf. [159] [21] [18]). Two publications address these issues and introduce methodologies for the development of valid BPMN extensions:

The methodology of Stroppi et al. [159] The authors introduce a methodological guideline for the development of valid BPMN extensions that keep conformance with the core BPMN elements. The methodology is based on approaches of Model-driven Architecture (MDA), using multiple model transformations. As a first step, the *Context Domain Model of the Extension (CDME)* is designed. New modeling concepts are integrated into existing BPMN concepts without the need of respecting the BPMN extension mechanism. In a second step, the CDME is translated into a *BPMN plus Extension (BPMN+X)* metamodel, annotated by different UML stereotypes. Transformation rules ensure conformity with the integrated BPMN extension mechanism, allowing the application of the BPMN+X model as a valid BPMN extension model. Afterward, interchangeability may be realized by automatically translating the BPMN+X model into XSDs using transformation rules and a provided conversion tool.

The methodology of Braun et al. [21] According to the authors, a weakness of the methodology provided by Stroppi et al. [159] is the missing analysis of the extension domain and its impacts on existing BPMN elements. Hence, a domain analysis is proposed, identifying the *requirements of the extension domain* regarding process modeling. Furthermore, an *equivalence check* is performed, comparing the requirements with existing BPMN modeling elements and their semantics. This prevents adding extension elements for requirements already satisfied by existing elements. Following these additional steps, the development process is continued with the steps described by Stroppi et al. [159].

2.2.3 BPMN Extensions

BPMN 2.0 has been extended in various ways since its release in 2011 [19] [180]. A majority of extensions address the domains healthcare, CPS, IoT, and ubiquitous computing. Others address the integration of business process resources, security aspects, and non-functional properties / Quality of Service (QoS) aspects (e.g., performance and reliability). Further on, specific extensions for domains such as cloud computing, manufacturing, disaster response, mobile devices, and knowledge management exist.

Conformity to the extension mechanism has raised in recent years. According to [19] less than 20 percent of available extensions implemented the provided mechanism up to 2014. As stated by [180], conformity has raised up to 48 percent in extensions published in recent years (period from 2014 to 2019). The low conformity rates are probably also caused by syntactical and methodical misunderstandings as argued by [18]. Valid extensions enable model interchangeability and remain compatible with the core of BPMN, an important aspect when using existing modeling tools.

This subsection briefly presents selected BPMN extensions with relevance for business processes in unreliable communication environments. Further on, the related work is distinguished from the approaches presented in this thesis. The extensions are categorized by their main objectives.

Integration of CPS, IoT, WSN, and Ubiquitous Computing

Many publications present solutions for the integration of CPS and IoT into the BPMN metamodel. Work concentrates on concepts for physical entities, sensors, and actuators, adding new task types, events, and attributes to the metamodel [113] [114] [81]. In [17], the authors of [13] extended their solution for CPS-aware resource modeling, realized by using *TextAnnotations* of BPMN. [51] aims at reducing the number of messages being exchanged with IoT devices by moving business logic to the devices. According to the approach, reducing communication efforts results in energy savings on IoT devices.

The work of [36] presents an overview regarding extensions for CPS and IoT. A comprehensive study of BPM in conjunction with IoT is provided by [34]. The authors state challenges and elaborate/compare numerous approaches published in the literature. The modeling of CPS using PMLs is the focus of [82], evaluating research publications regarding CPS requirements and the integration of non-functional properties such as time-related, physical, and behavioral aspects. While communication time is identified as a time-related property for CPS, unreliable communication is not considered. The survey identifies the need for a design-time verification of CPS behavior and related non-functional properties.

The IoT is often combined/used with WSNs. Modeling IoT device behavior in process models is part of [110] and [49]. BPMN models are firstly translated into a WSN neutral intermediate language and secondly into platform specific executable code. Afterward, the code is deployed to IoT devices for execution.

A BPMN extension to integrate WSNs is illustrated in [160]. The extension enhances the usage of sensor-based data by adding WSN tasks, WSN pools, and performance annotations. BPMN is used in [28], bringing together business analysts and software developers for modeling business processes and programming WSNs. WSN applications are designed in BPMN models, translated from the sequential semantic of BPMN into event-driven code for sensor network platforms. An integrated toolset facilitates the translation procedure and provides debugging functionality. Also considering code generation, [41] extends BPMN enabling process models to be transformed into executable code applicable for sensors and BPMN runtime engines.

Business process improvements based on the integration of ubiquitous computing are discussed in [176]. The proposed extension *uBPMN* enables the modeling of ubiquitous business processes by integrating tasks for sensing, reading, and collecting data and context information. With smart objects, environmental data collected by sensor/reader tasks becomes available for BPMN elements. As outlined in examples, integration of ubiquitous data allows improving business process performance, costs, and quality. In [177] and [179], the authors expand *uBPMN* for the integration of additional ubiquitous data (i.e., audio, images) and new modeling elements (e.g., sensing/reading events).

Also addressing ubiquitous computing in business processes, [53] introduces an extension to model mobile context-sensitive business processes. The extension provides new elements to define context groups and context events. Context is integrated using a context expression language, annotated to elements by *TextAnnotations*. Further on, activities may be marked with a context marker as context depending. The authors point out the need for research on code generation to ease the implementation/execution of process models. Based on this work, the domain specific modeling language *SenSoMod* [55] has been developed, including a corresponding modeling tool [56]. In [54], an evaluation regarding context modeling for mobile business processes is presented.

With CPS, IoT, WSN, and ubiquitous computing, major application domains exposed to unreliable connectivity are presented in this subsection. Most publications introduce new modeling elements to represent devices and their characteristics being part of the domains. While process models may benefit from meaningful graphical elements for the domains, connectivity is neglected in the extensions. No extension is identified that sets its focus on unreliable connectivity, providing new modeling elements

to strengthen process operation against connectivity issues. A verification of resilient operation at design time is missing.

Integration of Process Resources

Apart from CPS, IoT, WSNs, and ubiquitous computing, some publications address process resources and their integration into BPMN in a general manner.

A resource structure model is added to BPMN in [158]. Resource structure models characterize and classify resources, being usable in process models. The extensions' objective is to close the gap in communication and coordination between domain experts and technical experts part of process development.

[20] illustrates the development of an extension for resource-intensive processes in the area of mechanical engineering. New elements for human and non-human resources and their properties (i.e., skill, competence, knowledge of a human) are introduced. Resources may be allocated by activities and measured regarding cost and cycle time.

[16] allows specifying resources that are required by tasks to perform their operations. Resources are defined in a general meaning and may represent humans, software or hardware systems, and more. Based on their work on *pyBPMN* elaborated subsequently, the authors extend BPMN for complex resource modeling and management.

In [11], a BPMN extension for disaster response management is introduced. The focus lies on integrating comprehensive aspects regarding resources into process models. A resource may be human or non-human, consumable or non-consumable, has a location and a state, and can be single-usable or multi-usable.

The literature elaborates different approaches to include resources into process modeling. While there could have been relations to unreliable connectivity (e.g., in [11]), this is not the case. The approaches neither consider connectivity as a resource nor are they providing concepts applicable to guarantee/verify connectivity.

Integration of Performance/Reliability Aspects

Many of the publications address QoS in business processes. In the subsequently presented publications, QoS is referring to non-functional aspects such as performance, workload, and reliability of process tasks and resources.

Performance and reliability of BPMN processes are discussed in [13]. A metamodel extension named *Performability-Enabled BPMN (pyBPMN)* adds metadata descriptions about process/task performance characteristics (e.g., time required for a task) and reliability (e.g., mean time between failure for a task). Further elaborations in [15] apply simulations to verify process performance and reliability at design time. Both aspects are combined into a common verification metric called performability. Multiple

model transformations are applied until the simulation is performed. Performance and reliability in web service process models and SOA-based service discovery is part of [14]. The publication applies *pyBPMN* for web services.

Similar handling of process reliability is introduced in [150] and [52]. *relyBPMN* extends BPMN with a reliability value for activities. Using the stochastic workflow deduction method [29], the process is reduced iteratively until only one activity with a corresponding reliability value remains. The approach allows investigating process reliability at design time and at runtime.

The reliability of ambient assisted living systems is handled in [108]. By integrating reliability information about various used components into BPMN, the overall reliability can be evaluated for appropriate resource allocation. In [50], the authors extended their work to not only integrate reliability information of tasks but reliability information of resources used by tasks to realize their operations. Effects of human and non-human resources on process reliability are considered.

In [112], a mechanism based on semantically annotated process models allows compensating faulty tasks for process service plan executions in cloud environments.

riskaBPMN, an extension for quantitative risk assessment, is introduced in [30]. Related to the work of [52], BPMN is extended for values of a risk likelihood and the consequence of that risk in terms of a business loss value. While the publication integrates these values into BPMN, the required methods/calculations to state process risks are considered as future work by the authors.

The presented extensions address non-functional properties of business processes, mostly related to activities and resources. While some extensions are limited to adding new QoS attributes [30], other elaborate methods verifying the chosen QoS parameters [15] [52]. However, QoS aspects like bandwidth, latency, and failure probability of the transporting network as well as the requirements of messages and are not considered. Metrics measuring resilient operation are missing.

Integration of Flexibility Aspects

Integration of Quality of Information (QoI) and access cost of IoT resources into business processes is addressed in [106]. The BPMN task element is extended with attributes for a QoI metric and value as well as an access cost metric and value. Examples of access costs are consumed energy, consumed bandwidth, monetary costs. A business process gains flexibility in selecting adequate IoT resources with the help of a briefly sketched IoT middleware. The middleware can compare the required minimum QoI with the maximum access costs of available sensors dynamically at runtime, selecting an appropriate resource.

[107] introduces *CF4BPMN*, an extension for controlled flexibility in business processes. The extension enables domain experts to state which, where and how process elements may be changed. Defined expressions about possible changes are added using *TextAnnotations* to corresponding elements. Elements allowed to be changed are colored in green in process models. Models may be changed at design time.

Process flexibility in collaborative processes between participants of different organizations is addressed in [152] and [153]. Flexibility is realized by adding a versioning option for modeling elements, such as participants, message flows, tasks, events, sequence flows, and gateways. If required, flexibility may be obtained by adding/deleting elements from process models and specifying a new version.

In [35], an algorithm for the multi-criteria-based selection of web services is introduced. First, the BPMN process model is translated into a business process ontology, as the web services of a service registry are translated into a web service ontology. Afterward, the instance of the process ontology is matched with the instances of the web service ontologies. Selection is based on functional and non-functional properties.

Process variability in terms of different process configurations is addressed in [178]. As examples for process variability the authors state the production of hamburgers with different ingredients (i.e., slice of bacon/cheese, lettuce, buns, sauce) and the different configurations of airplanes in a common production line. To avoid managing different process models for similar, but slightly different workflows, a common model including all variable configurations is modeled. Using variability patterns, a specific model for a certain configuration may be generated based on the common process model.

Flexibility can be seen as a tool supporting optimal operation of business processes. The ability to model optimal operation under the given conditions is beneficial for processes taking place in unreliable communication environments. Since connectivity changes dynamically, approaches concentrating on design time flexibility are inadequate [107] [152] [153]. Further on, it is necessary to verify optimal process operation as the sum of all decision points part of the process, not only of a local/isolated decision point (missed by [35]). In addition, none of the extensions specifically addresses unreliable communication by providing alternatives for failing message flows.

2.2.4 Modeling a Slurry Application

This subsection evaluates the modeling of business processes using BPMN 2.0.2 to elaborate to what extent BPMN is capable of modeling unreliable communication. An environmental-friendly slurry process is modeled and evaluated regarding its suitability for unreliable environments. The subsection concludes by presenting limitations of using BPMN in unreliable communication environments.

Figure 2.3 shows the slurry process *S1*, modeled in BPMN. *S1* consists of three parts: Part *a)* considers the creation of an AppMap, defining partfields and the application of slurry ingredients (e.g., nitrogen, phosphorus). Part *b)* defines how the slurry is analyzed for its ingredients. The three alternatives *Laboratory (LAB)*, *Near-Infrared Spectroscopy Sensor (NIRS)*, and an *Ingredients Reference Service (REF)* are defined. *Analyze slurry* represents a sub-process, encapsulating the details of choosing one of the alternatives. In part *c)* the slurry is applied to the field using *GPS*.

Considering communication resilience, a domain expert is unable to evaluate whether or not *S1* will operate resiliently. The functionality of all three parts is realized by calling services at remote participants. In the case of intermittent or broken connectivity, the message flows between *S1* on the slurry spreader and the remote participants may be significantly delayed or non-existing. Parts *a)* and *b)* of *S1* require cellular communication, while the position is received by *GPS*. Depending on the area, cellular communication may face connectivity issues. By not communicating with one of the remote participants, the process may operate incorrectly or can completely break down. Plants may not grow as expected, resulting in a loss of yield. Incorrect slurry applications violating legal requirements may end up in penalties.

Further on, only one of the three alternatives *LAB*, *NIRS*, and *REF* of part *b)* needs to be available for resilient operation. However, based on BPMN modeling elements, these details are hidden by the sub-process *Analyze slurry*.

The limitations in modeling resilient processes using BPMN are illustrated by the following three examples. The slurry processes *S-ERR*, *S-GW*, and *S-DMN* show models for the sub-process *Analyze slurry* of Figure 2.3. The applied modeling elements and the structure of each sub-process differ.

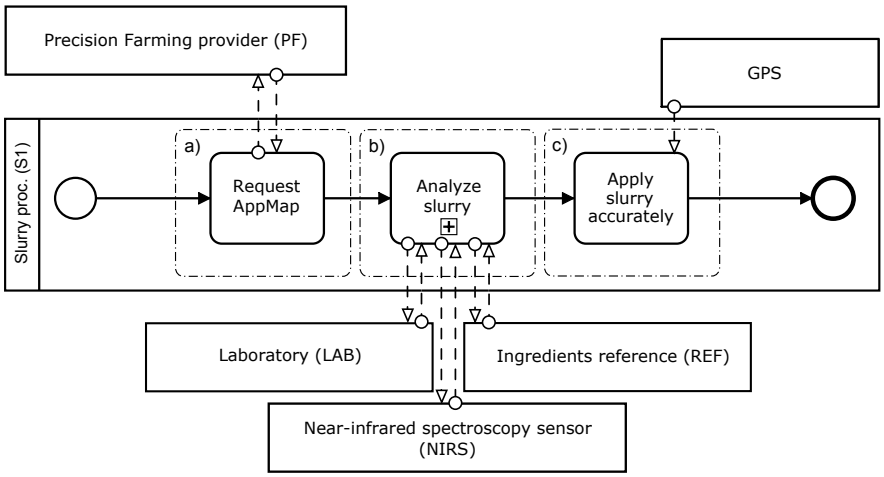

Figure 2.3: Slurry process *S1* (BPMN).

Process *S-ERR* presented in Figure 2.4 uses error event elements to model alternatives for failing communication segments. First, it is tried to request an analysis report from a *LAB*. If this fails due to connectivity issues or no available data for the current slurry job at the *LAB*, the error event (circle with flash icon) is executed referring to the second option *NIRS*. If communication to *NIRS* is unsuccessful (e.g., since there is no *NIRS* on the slurry spreader), another error event points the execution to the final option of calling *REF*. If this call also fails due to connectivity issues, the sub-process is directed to end in a failure state.

All three alternatives are handled in a fixed priority, one after another. Although timeouts are not modeled, this solution approach requires some modeling effort. Triggering an error event requires programming code linked to the corresponding task, which is hidden by the model here. Besides, the error events for *LAB* and *NIRS* encapsulate different causes: At *LAB*; the error is triggered if no connectivity is available to *LAB* as well as if *LAB* has no data for the current job. Adding alternatives by using error events complicates the expressiveness of the model and its functionality. Changes in the prioritized order require additional modeling effort. Hence, the dynamic adaption of priorities during runtime is not possible. In cases of no connectivity, the sub-process ends in a failure and will break down the complete slurry process.

Fixed priorities are eliminated by the second modeling approach using BPMN, illustrated in Figure 2.5. In *S-GW*, an exclusive gateway is used to decide on an analysis option. Depending on the value of the process variable x, an option is chosen. x is configured with an initial value at the start of the process and can be changed during execution. If connectivity with a participant fails, an error event is triggered

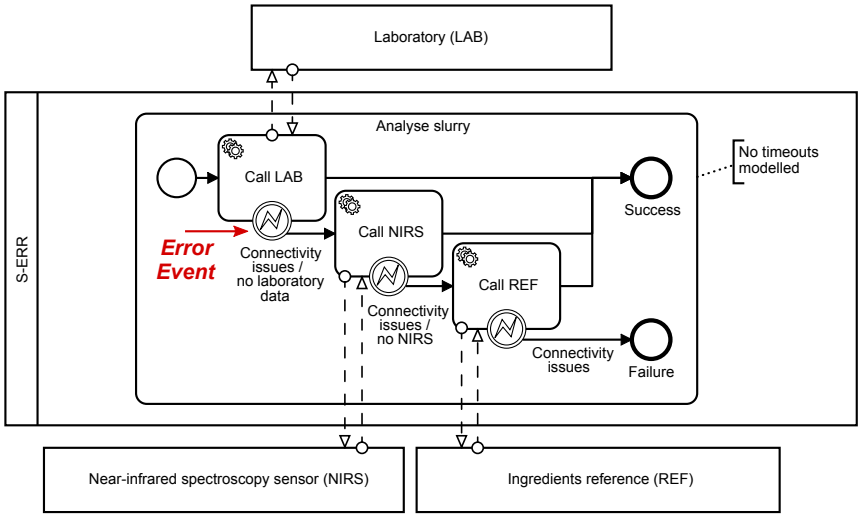

Figure 2.4: Modeling of a resilient process using BPMN error events (*S-ERR*).

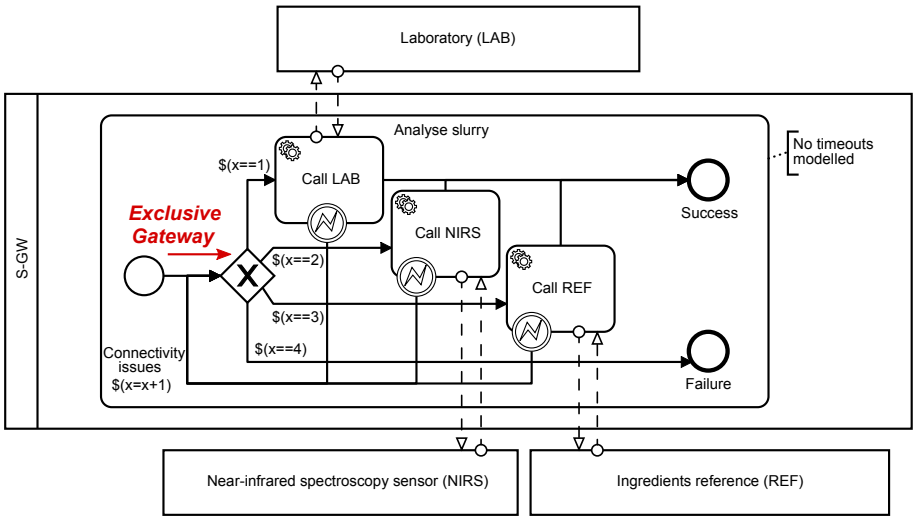

Figure 2.5: Modeling of a resilient process using a BPMN exclusive gateway (*S-GW*).

and x is assigned a new value calling another option. While the preference for one option or another can be changed to the scenarios' demands, the sub-process may still end in a failure. Besides, additional options for a slurry analysis provided by dynamically appearing participants remain unused.

The third modeling approach *S-DMN* in Figure 2.6 utilizes a business rule task and adds fewer graphical elements to the model. The business rule task is represented by a DMN decision table, evaluating inputs using a set of rules to outputs. However, no set of inputs needs to be evaluated against a static rule set here. Instead, it is required to compare dynamic alternatives with each other, choosing the most appropriate alternative to be invoked by a call activity. Figure 2.7 depicts a DMN decision table realizing the business rule task of *S-DMN*. The two process variables *Precision* and *Cost* decide for one of the alternatives, based on their values. Although this principle adds flexi-

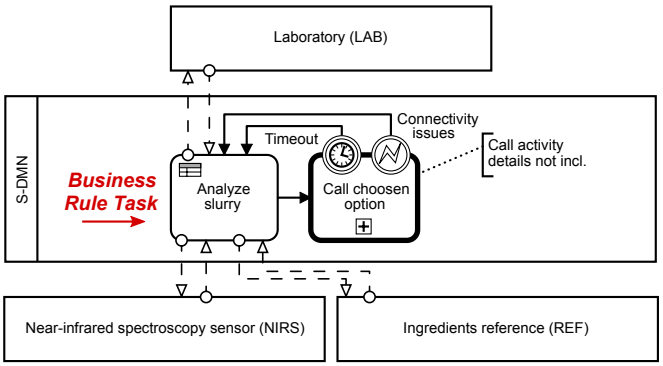

Figure 2.6: Modeling of a resilient process using a BPMN business rule task (*S-DMN*).

Analyze slurry | Hit Policy: First

	When	And	Then	
	Precision (string)	Cost (integer)	Analysis (string)	Annotations
1	high	>= 80	call-LAB	If high cost is okay.
2	high	< 80	call-NIRS	If cost shall be moderate.
3	low	-	call-REF	If there is no need for high precision.
+	-	-		

Figure 2.7: Implementation of a business rule task using the Decision Model and Notation (*S-DMN*).

bility to adapt the decision based on multiple criteria to the scenarios' demands, it is not possible to include the current connectivity conditions into the decision. Hence, *S-DMN* is unable to guarantee resilient process execution.

Table 2.1 compares and evaluates the three implementations of the Analyze slurry sub-process, modeled in BPMN. The examples are missing the required flexibility to address scenarios with varying slurry analysis demands and connectivity characteristics. While some may operate well in certain scenarios, they may be inadequate for other scenarios. This results in adapting process models for different scenarios and maintaining the models continuously.

Since maintaining process operation in case of connectivity issues and process dynamics may be challenging, every possibility of preventing interruptions and breakdowns should be considered. While required participants may be missing, other participants may be offering equivalent functionalities and may be used as a replacement.

Achieving the desired process resilience is often challenging for domain experts. While they want to focus on technical aspects of their process, they have to consider and balance the use of available modeling elements and adapt the process structure accordingly. Transparency and clarity of models may suffer. Expert-level modeling skills and programming of code may be required.

Besides the mentioned modeling limitations, no method for the verification of process resilience at design time is provided. Domain experts are unable to verify and optimize the resilience of their process models. Error-prone and failing process executions are the consequence.

Overall, the performance of BPMN regarding the modeling of resilient processes is constrained.

Table 2.1: Comparison of modeling approaches for resilient processes using BPMN.

Aspect	*S-ERR*	*S-GW*	*S-DMN*
Use of existing BPMN / DMN elements.	+	+	+
Graphical expressiveness of unreliable connectivity, alternatives, and priorities.	o	o	-
Avoidance of additional graphical hierarchies for the modeling of unreliable communication characteristics.	o	o	o
Ability to visually indicate possibly failing message flows and to integrate the corresponding QoS requirements / connectivity-characteristics.	-	-	-
Ability to adapt operation for unplanned runtime dynamics (e.g., fluctuating connectivity, appearing/disappearing participants).	-	-	-
Ability of the model to adapt for the operation demands and connectivity characteristics of different scenarios.	-	o	o
Ability to continue process operation in case of no connectivity.	-	-	-
Simplicity of modeling resilient processes.	-	-	-
Ability to verify process resilience before runtime.	-	-	-
Summarized points	4	5	4

Declaration: + ⇒ full support / 2 points,
o ⇒ limited support / 1 point, - ⇒ no support / 0 points

2.2.5 Findings and Remarks

When modeling a process taking place in unreliable environments using BPMN, a significant part of the process model is dedicated to handling communication failures. Alternatives for possibly unavailable message flows have to be added, but can not be identified as alternatives. Using the same process model in different scenarios may require model changes due to missing flexibility. Domain experts are forced to extensively address communication aspects and lose focus on the actual objectives of the process scenario. Eventually, domain experts may get stuck at a point where BPMN fails to address a communication-related issue. Domain experts are unable to verify resilient and optimal operation of a modeled process. This may result in process failures or breakdowns at process runtime.

To the best of the authors' knowledge, none of the available BPMN extensions focuses on modeling resilient processes for unreliable environments, none features a resilience verification mechanism. The solutions do not address the integration of communication requirements and scenario-based connectivity descriptions into BPMN.

Straightforward mechanisms for the modeling of communication-issue-related alternatives by domain experts are missing.

2.3 Analysis of Business Processes

Analyzing and optimizing business processes represents a major challenge in BPM. It is a broad field of research where different concepts and approaches are applied.

An example of ongoing research activities is represented by the area of process mining, capable of enhancing business processes. The topic combines process modeling and analysis with computational intelligence and data mining techniques (cf. [166] and [167]). Process mining may identify comprehensive aspects and correlations of executed business processes and included parameters. However, the application of mining techniques requires the existence of datasets about past process executions. This is not always the case, especially for newly designed process models ready to be verified for resilient and optimal process operation.

Numerous theoretical and practical approaches for the analysis of processes are available, not all being applicable at process runtime (cf. [61]). Other approaches are integrated into BPM software suits as commercial products, not being usable with custom PML extensions. For instance, [27] enables the analysis of processes regarding bottlenecks, automation, and data-driven improvements.

Graphs have the potential to address challenges 2 and 3 of this thesis, namely verifying resilient and optimal process operation. Diverse graph algorithms are outlined in the literature, solving single-criterion and multi-criteria optimization problems. The algorithms may be applied at process design time and process runtime. Numerous open source implementations are available for many algorithms, not limiting their use to specific application domains or commercial products.

Subsequently, fundamentals and related work regarding the single-criterion and multi-criteria graph-based decision-making are presented. Following, related work regarding the usage of graphs in conjunction with BPMN is reviewed.

2.3.1 Graph-based Decision-Making

A graph represents a structure of vertices and edges. Edges may be directed or undirected and are used to connect vertices with each other. In general, a graph G is defined as $G = (V, E)$ with V as a set of vertices and E as a set of edges connecting vertices. With $V = \{a, b, c, d\}$ and $E = \{(a, b), (b, d), (a, c), (c, d)\}$, a graph with four vertices and four edges is described. Figure 2.8 illustrates the graph with circles as vertices and lines as edges. Since the edges have no directions, they may be traversed from both connected vertices. The graph is defined as an undirected graph.

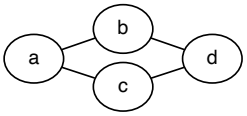

Figure 2.8: An undirected graph.

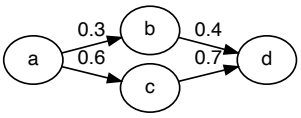

Figure 2.9: A weighted, directed, and acyclic graph.

By adding a direction to an edge, the edge is only allowed to be traversed towards its defined direction. If there are no loops within the graph, the graph is defined as a *Directed Acyclic Graph (DAG)*. Further on, adding weight values to edges allows specifying a criterion, such as cost. Figure 2.9 illustrates a weighted DAG with arrows as directed edges and attached values as edge weights.

Since graph structures are capable of addressing various problem statements, they are often applied for decision-making. One of the most popular problems is defined as finding the shortest path in terms of cost between two vertices (i.e., decide on one of many paths). Determining the shortest path may be an approach for the resilience analysis discussed in this thesis.

A solution was first introduced by the *shortest-path-first (SPF)* algorithm of Bellman-Ford [10], followed by an SPF algorithm of Dijkstra [65]. The work of [59] diversified the problem, e.g., by considering the shortest paths between all vertex pairs of a graph and identifying the second, third, etc. shortest graph path. Since operating an SPF algorithm on large graphs with thousands or even millions of vertices and edges requires a reasonable amount of computation work, many different SPF speedup techniques exist. The *A* algorithm* [85] aims at reducing the computational effort of an SPF graph search by prioritizing vertices that will probably lead fastest to the destination vertex. Identification of the most promising vertices is realized by heuristic functions, different functions are available. *Contraction Hierarchies (CH)* represent another speedup technique. Often used in large graphs of road networks, CH try to identify shortcuts on the way to the destination vertex [77]. Therefore, the importance of vertices is evaluated in a preprocessing phase. Afterward, a graph search skips vertices which are less important. A comparison of different speedup techniques is provided by [8].

An SPF search may include graph edges not desired for the shortest path. Hence, algorithms for a *constrained SPF (CSPF)* include constraints regarding applicable edges for the graph search [183]. Unqualified edges are not chosen for the path between the

source and destination vertex. In addition, some scenarios have interest in the second, third, etc. shortest path. This can be identified by using algorithms for the *k-shortest-path (KSP)* [63]. Other algorithms consider the dynamic updating of graph edge weights and the re-computation of the SPF [32] [22] [175].

Another problem domain of graph algorithms is denoted by *network graphs*, featuring a capacity criterion as edge weights [4]. The maximum throughput between a pair of vertices in a network graph is identified by maximum flow algorithms. Additionally, a cost criterion may be added to determine the maximum flow at minimum cost.

The reader is referred to the literature for further information on graphs and algorithms. For example, background information may be found in [65] and [64].

2.3.2 Graph-based Multi-Criteria Decision-Making

The majority of graph algorithms presented in the previous subsection consider a single criterion, such as cost, for optimization. Since many use cases include more than one criterion, algorithms for *multi-criteria/multi-objective decision-making* evolved. These algorithms may be suitable for addressing challenge 3 of this thesis, the multi-criteria process operation.

One of the first approaches in this area is [84], providing algorithms for the multi-criteria SPF problem. Other algorithms solve the multi-criteria problem by finding a set of *Pareto-optimal* results. A graph path is defined as Pareto-optimal if it is not possible to optimize one criterion without deteriorating another criterion. Some of the first approaches for finding Pareto-optimal sets include [39] and [109], while [169] allows speeding up computation time by estimating Pareto-optimal paths. Again, a variety of publications focus on large graphs. [47] is applicable for train networks, where optimal paths considering travel time, number of train changes, and reliability of transfers are found. Also addressing public transportation, [43] introduces a round-based speedup technique for dynamic networks avoiding the need for preprocessing. The work of [42] and [9] uses a multi-criteria optimized version of Dijkstra for finding Pareto-optimal graph paths. A technique is provided to reduce the number of considerable paths. Some publications consider constraints in multi-criteria optimization [147] [57] [102] [155].

Several algorithms apply *scalarization methods* to reduce multiple criteria to a single criterion. Using scalarization methods such as the *Weighted Sum Model (WSM)* allows combining weights of different criteria to a scalar graph edge. For instance, [87] combines WSM with Dijkstra to solve the multi-criteria optimization. In [33], WSM is applied to optimize energy savings. Scalarization techniques allow combining criteria in [151], enabling energy-aware scheduling of machines in manufacturing.

Overviews of the different concepts for multi-criteria optimization are presented in [139] and [40].

2.3.3 Graphs and BPMN

BPMN is sometimes described as a graph-like language or a language based on graph structures. Considering directed graphs, there are few obstacles in translating a process into a graph. More challenges arise if a DAG is required since repetitions of process segments have to be addressed.

[45] introduce formal semantics in terms of translating BPMN to Petri nets, allowing to apply existing analysis methods to process definitions. The authors continue their work in [44] to check BPMN processes for similarities concerning tasks and control flows.

[31] uses Bayesian Networks to advise on human activities. The process model has to be in a BPMN normal form (i.e., no directed cycle or loop) to be translated into so-called agents, represented by a DAG. The work of [62] is used for process mapping, respecting the characteristics of agents (featuring plans, goals, intentions, beliefs).

Being motivated by optimization techniques for data-intensive queries and flows in data management, [80] illustrates a concept for automated performance optimization of BPMN processes. The objective is to apply techniques used in big data analyses in business processes for optimization purposes. A minimization of performance cost (i.e., execution time) is intended by reordering and paralyzing (ad-hoc) process tasks using existing algorithms for data-intensive queries, working on DAGs. For this purpose, the BPMN model is translated into a DAG. The vertices are annotated with statistical cost and metadata information of process logs, forming the basis for optimizing the average cost of multiple process executions. The outcome is an optimized, reordered execution plan, which needs to be integrated back into the process model for execution. The concept is further elaborated in [99], now focusing on the reordering of tasks for performance optimization. Additionally, [163] and [164] evaluate process models for their eligibility for redesign to apply data flow optimization techniques. In other words, the models are checked for their reordering capabilities. Not specifically addressing BPMN, an approach to analyze the response time of data flows executed in parallel and distributed environments is introduced in [98]. The response time includes communication cost in terms of time to transfer data between machines. The objective is again the reordering of tasks.

[116] provides process optimization based on structural causal models. A framework for answering *what-if* questions about processes is illustrated. DAGs are used to realize the structural causal models applied for reasoning.

[141] applies segmentation and restructuring to business processes for workflow scheduling and operation in edge-cloud environments. This allows splitting a process into multiple sub-processes, being distributed across different edge clouds. The authors map BPMN models to DAGs to apply a segmentation method. Mapping is realized by

a proposed algorithm, translating activities to vertices and sequence flows to edges. As shown in the case verification, the translation algorithm is rather simple and limited: the BPMN process model only consists of activities and exclusive gateways. Translation of parallel/inclusive gateways, events, message flows, and loops are missing/unfeasible.

2.3.4 Findings and Remarks

A lack of concepts translating process models into graphs with respect to communication resilience and other criteria is determined. Metrics to measure, rank, and verify resilient and optimal operation are missing. With a translated graph on hand, it has to be evaluated how a graph has to be prepared for the application of graph algorithms and which algorithms are suitable. For instance, many multi-criteria graph algorithms identify sets of Pareto-optimal graph paths. However, the suitability of Pareto-optimal paths for multi-criteria process optimization including communication resilience is unconfirmed.

Some publications in the literature translate process models into graph structures for various reasons. While most of them translate activities to vertices and sequence flows to edges [45] [44] [31] [80], translation of other process elements such as segments including parallel gateways and loops differs. Compared to each other, the resulting graphs have different meanings and are designed for a different purpose. For instance, comparing this thesis with [80] outlines differences in the graph translation, its meaning, and its analysis using graph algorithms. While [80] tries to reorder and paralyze tasks, this work determines resilient and optimal operating process paths. Approaches for the verification of resilient and optimal process operation at design and at runtime are missing in the literature.

2.4 Execution of Business Processes

This section outlines the fundamentals of business process execution. The fundamentals are relevant regarding challenge 4 of this thesis, resilient process execution. At first, selected tools supporting the implementation and execution are introduced. Following, the resilient execution of the slurry application process *S1* is considered before a summary of findings and remarks concludes this section.

2.4.1 Tool Support for Process Execution

Software suites for BPM are collections of tools for modeling and executing business processes. Many suites combine modeling tools, runtime engines for the execution of process models, monitoring and controlling tools. The software tools are often bundled

with services for support and analysis of business processes, available for purchase as commercial products. However, not every PML can execute modeled processes. This was also the case for the first version of BPMN, released in 2006.

Since BPMN didn't specify execution semantics in its initial release 1.0 [123], it was often combined with *WS-BPEL* [122] and *XPDL* [173] for the execution of processes. The BPMN standard even includes guidelines for translating a BPMN model into an executable WS-BPEL process ([125] p. 445ff.). Starting with BPMN version 2.0 and the rise of BPMN execution engines, usage of WS-BPEL and XPDL for process execution lost relevance. Numerous runtime engines are available on the market, capable of executing processes modeled in BPMN. Some of the more popular engines include Activiti [2], jBPM [145], Signavio [156], and Camunda [25]. This thesis employs Camunda as the exemplary runtime engine.

However, the execution of process models requires additional effort. Most runtime engines only support a subset of the BPMN modeling palette [75] [76]. Camunda realizes the execution of process models by extending models for custom elements, providing execution information processed by the runtime engine. So-called *delegate classes* allow linking activities with custom code for their execution. Detailed information about the extension elements and supported modeling concepts can be found in the Camunda Platform Manual [26].

While Camunda is implemented in Java, different programming languages and techniques may be used to program process functionality or to integrate remote services. The Camunda REST-API allows to monitor, control, and adapt process execution using REST-calls. By using delegation code, flexibility to integrate any other language or technology is provided. Furthermore, expressions, connectors, scripts, and external tasks may be used instead of delegate classes.

Usually, the functionality of activities represents the business logic of a process. While it may be implemented as part of the process module executed within a runtime engine, many processes include remote services to realize business logic. The architectural principle of microservices gained popularity in realizing such remote services over recent years. Microservices implement and offer functionality while minimizing dependencies to other services [117]. A microservice may be seen as a self-contained service for a specific purpose, being combined with other microservices to build up applications providing a broad range of functionality. One approach to orchestrate an application containing multiple microservices is to use BPMN. Within activities of an orchestration process, the specific microservices are called. This way, the process itself is defining which microservices are used in what exact order, and how results are interpreted in decision points such as gateways [24].

2.4.2 Executing a Slurry Application

While BPMN runtime engines such as Camunda are often used in cloud environments, no restrictions regarding their use in unreliable communication environments exist. Connectivity issues will be identified at the network layer, running processes will notice them as timeouts and failures when communicating with other participants. However, existing tools used in cloud environments have not been designed for various kinds of connectivity issues. Repeating communication delays and interruptions to service offering nodes often lead to their long-term removal from service registries. An example of such a mechanism is the *circuit-breaker pattern* [71], which automatically excludes services showing connectivity issues from service registries. However, these services may operate well during specific time frames of a process. Due to missing alternatives in unreliable communication environments, this may affect process operation negatively.

For the execution of the slurry application process modeled in section 2.2.4 (p. 25ff.), code implementations for the process *S1* and the three *Analyze slurry* subprocess variants *S-ERR*, *S-GW*, and *S-BR* have to be developed. With linked business logic implementations, the models can be executed in a BPMN runtime engine such as Camunda. However, a resilient execution of the processes can neither be verified at design time nor be guaranteed at runtime.

The problem of resilient process execution rises with the elaboration and integration of solution approaches for the resilient and optimal operation into process models (challenges 1 to 3 in Figure 1.1, p. 7). Process execution has to address and implement the corresponding approaches.

2.4.3 Findings and Remarks

Comprehensive tool support for the implementation and execution of process models exists. However, the tools are not designed for unreliable connectivity on a large scale as it happens in unreliable communication environments. While various communication techniques applicable for unreliable communication environments exist, their combination with business processes needs to be addressed. Process execution needs to implement the approaches for resilient and optimal operation defined in chapters 3 to 5 (p. 41ff.). Hence, an analysis in chapter 6 (p. 123ff.) determines the requirements for process execution.

Summary

This chapter presents the foundations and related work for the modeling and execution of resilient business processes. Characteristics of unreliable communication environ-

ments and their impact on processes are illustrated. Connectivity issues affecting process communication with collaborative participants is identified as a major challenge for process operation. A literature review of available BPMN extensions determines open research issues regarding processes facing delayed, intermittent, or broken connectivity. Modeling of the slurry application process shows that BPMN and its extensions fail to satisfy challenge 1, the modeling of resilient processes. No mechanism for the verification of resilient and multi-criteria process operation exists (challenge 2 and 3). Although comprehensive support for process execution exists, a resilient operation is not ensured (challenge 4). Since the issues of process execution are affected by the solution approaches for challenges 1 to 3, these issues are identified and addressed in chapter 6 (p. 123ff.).

CHAPTER 3

MODELING OF RESILIENT PROCESSES

The ability to adapt to connectivity-related issues is of key importance for the resilient operation of processes. The attempt to model a resilient slurry process in section 2.2.4 (p. 25ff.) reveals limitations of BPMN and its modeling palette. The different approaches to model the *Analyze slurry* sub-process miss flexibility required to address varying scenario characteristics, are vulnerable regarding connectivity issues, lack a method to verify resilient operation, and may end up in breaking process executions at runtime. Further on, none of the available BPMN extensions fills this gap by addressing processes in unreliable communication environments.

This chapter introduces *resilient BPMN (rBPMN)*, a BPMN extension for business processes in unreliable communication environments. Strategies for the modeling of resilient processes are defined. New modeling concepts are introduced, realizing the resilience strategies and resolving the remaining set of unfulfilled requirements.

While BPMN includes its own extension mechanism, many extensions published in the literature miss conformity with this mechanism [19] [180]. Hence, an MDA-based approach for the development of valid BPMN extensions was introduced by [159] and enhanced by [20]. The enhanced approach is used as a methodology to develop *rBPMN* in this thesis. The development procedure of *rBPMN* is illustrated in Figure 3.1. The manual tasks (hand symbol in the topper left corner) specify manual developments steps, while service tasks (gearwheels in the topper left corner) show automated steps. Following the recommendations of [20], the domain of unreliable communication environments is first analyzed for its *requirements* regarding a PML. An

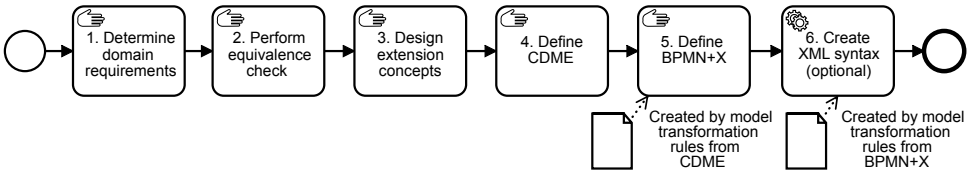

Figure 3.1: Development procedure of the BPMN extension *rBPMN* (modeled in BPMN).

equivalence check identifies requirements that are satisfied by existing BPMN concepts. In accordance with [159], new modeling concepts and a *Context Domain Model of the Extension (CDME)* are developed, resolving unsatisfied modeling requirements. The development procedure concludes by applying two model transformations: At first, the CDME is transformed into a *BPMN plus Extension (BPMN+X)* metamodel, followed by a transformation into a *BPMN XML schema*. Since the development steps 5 and 6 are simply applied using transformation rules, the focus of this thesis lies on steps 1 to 4 (cf. Figure 3.1).

This thesis makes use of the terms *modeling element* and *modeling concept*. Modeling elements are defined as the objects present in a process model, used to define the process workflow. Modeling concepts define the semantics and syntax of modeling elements in metamodels. In this regard, the term *metamodel* refers to the CDME, the BPMN+X model and the BPMN metamodel. While the purpose of modeling concepts and modeling elements differs, their semantics regarding the use in process modeling are equivalent.

In the following sections the development of *rBPMN* is described according to the development procedure of Figure 3.1. The chapter concludes by presenting a resilience verification method for individual message flows, using message flow properties, QoS requirements, and connectivity properties part of an *rBPMN* process model.

3.1 Domain Requirements

In the first step of the extension development, unreliable communication environments are analyzed regarding their modeling requirements. Resilient operation of processes is required by scenarios of different application domains. While the degree of process dynamics and participant collaboration varies, all domains face an uncertainty of connectivity resulting in vulnerable and failing processes.

The subsequent paragraphs include statements describing different aspects of business processes. The aspects cover necessary abilities for general and resilient process modeling. Directly following each paragraph, the essential points are summarized in

one or more requirements (abbreviated: req.). The determined requirements represent demands to be met by the new BPMN extension. To address a broad range of scenarios and application domains, requirements are determined in a generally applicable manner.

Processes need to be defined, organized, controlled, monitored, adapted, and documented. Many processes require the collaboration of different participants (e.g., human activities, mobile machinery, digital services, different organizations). While collaboration is required, modeling shall be dividable, reusable, and respect trade secrets. This leads to the following requirements:

Req. 1: Ability to model collaborative processes including different participants.

Req. 2: Ability to divide collaborative work into reusable subsegments and to model them as black boxes.

The work needs to be controlled and monitored. Autonomous and manual adaptions may be required, e.g., for automatically adapting machine parameters or manually deciding on machine failure replacements. This concludes to the following requirements:

Req. 3: Ability to adapt processes automatically based on predefined variables/events.

Req. 4: Ability to manually adapt running processes based on process state / proposed solutions.

Many scenarios take place in rural, sparsely populated environments with a lack of cellular coverage. Other scenarios may include performance-, communication-, or energy-restricted devices. This causes delayed, intermittent, rapidly changing, or even broken connectivity between the scenarios' participants. This results in the following requirements:

Req. 5: Ability to define Quality of Service (QoS) requirements for process communication and characteristics of participants' connectivity.

Req. 6: Ability to ensure resilient operation even during intermittent or broken communication between participants.

Identification of and adaptation for an optimal process operation are required, especially under challenging communication conditions. Technical possibilities to evaluate and optimize the process at design time are needed to lower the risk of process failures at runtime. This leads to the following requirements:

Req. 7: Ability to explicitly model optimal process operation at design time.

Req. 8: Ability to identify optimal process operation in a dynamically changing scenario at runtime.

Req. 9: Ability to verify process resilience for a given scenario at design time.

Based on the variety of different scenarios and application domains, not every requirement is of importance for every single process. Hence, an indirect requirement with practical relevance arises for the BPMN extension: A *lightweight implementation* of requirements is desired. In the context of this thesis, a lightweight implementation is defined as the possibility to implement only a subset of requirements and corresponding modeling concepts.

The definition of a process model and its execution are closely linked to each other. The model defines the scope, semantics, structure, and rules to be followed during execution at process runtime. While this chapter outlines process modeling using concepts and elements for resilient operation, chapter 6 (p. 123ff.) considers the realization of the *resilience strategies* introduced by the model during process execution.

3.2 Equivalence Check

Depending on the PML used to design a process, some of the requirements for resilient process operation may be already fulfilled. An equivalence check represents the second step of extension development (cf. Figure 3.1). It allows the determination of requirements that are already satisfied by existing modeling concepts of the utilized PML.

Subsequently, an equivalence check is performed to compare existing concepts of the BPMN metamodel with the requirements for the modeling of resilient processes taking place in unreliable communication environments. First, general process modeling aspects are considered in Table 3.1. The *Support Level (SL)* is defined as the evaluation criterion, stating how well existing modeling concepts are addressing a requirement. SL differentiates between + ⇒ full support, o ⇒ limited support, and - ⇒ no support. Explanations of the different modeling concepts and their semantics are part of the BPMN specification ([125] p. 26ff.).

The required modeling concepts for dividing a process into a connected flow of different steps are provided by BPMN. BPMN activities (e.g., task, service, script) represent process steps, connected to each other with sequence flows. The flow can be controlled by different types of gateways using process variables. However, modeling gateways with the necessary flexibility is often challenging due to the dynamic nature of processes. Using sub-processes, activities may be hierarchically ordered and reused. Integration of user-controlled tasks allows adapting process operation by humans.

Table 3.1: Equivalence check of BPMN concepts for general process modeling aspects.

Req.	**Element**	**Semantics** (and support explanation, where applicable)	**SL**
		General process modeling	
1, 2	Activity / Task	Part / step of a process.	+
1, 2	Process	Reusable container for a set / flow of chosen activities.	+
1, 2	Sub-process Call activity	Encapsulates / hides activities in processes, allows hierarchy levels, allows reuse.	+
1	Sequence flow	Coordinates the process flow.	+
3	Gateway Business rule task	Allows autonomous process flow decisions based on defined variables. Limited support: not designed for handling dynamics based on unreliable environments.	o
4	User task Manual task Ad-hoc sub-proc.	Allows user-based decisions for non-automated situations.	+
3, 4	Event Signal Conditional sequence flow Timer	Allows to react (dynamically) on messages, events, conditions, timings. Allows to dynamically create process instances / configurations.	+
1, 2	Text annotation	Specifies descriptive information, no impact on sequence flow.	+

Support-Level (SL): + ⇒ full support, o ⇒ limited support, - ⇒ no support

Aspects of collaboration, communication, dynamics, and decision modeling are analyzed in Table 3.2 and outlined subsequently.

Pools and participants enable collaboration with different participants and system parts. By using collapsed pools, no process internals and trade secrets have to be shared with other participants. In terms of mobility, participants may be stationary or mobile.

BPMN provides only limited support for the modeling of communication. While message flows allow the modeling of communication between participants, there is no possibility to declare a group of message flows as a set of alternatives in case of connectivity issues. Not every message flow may be required for the resilient operation of a process. For instance, log messages to other participants are often not critical for process resilience. However, required and optional message flows can not be distinguished regarding their importance for resilient operation. Some scenarios use a message item definition as a graphical, named concept representing a message exchanged between participants. Although it may contain the message structure, information about the size, frequency, and relevance of the message is missing. BPMN provides no support for the definition of message QoS and scenario connectivity (req. 5) as well as for the resilience verification of process models (req. 9).

Table 3.2: Equivalence check of BPMN concepts for the modeling of collaboration, communication, dynamics, and decisions.

Req.	**Element**	**Semantics** (and support explanation, where applicable)	**SL**
		Collaboration modeling	
1	Participant Pool	Defines / structures different participants.	+
1, 2	Collapsed pool	Defines different participants, hides internals in black boxes.	+
1	Pool lanes	Separation of concerns / organization of activities within a participant.	+
		Communication modeling	
1	Message flow	Communication with other participants. Limited support: no possibility to specify alternative message flows / to specify flows as optional based on connectivity.	o
1	Message ItemDef.	Graphical concept and name for a message. Limited support: frequency, size, relevance not included. Message ItemDef. not widely used in executable BPMN environments.	o
5, 7, 8, 9	Not available	No support: no BPMN concept fulfills requirements 5, 7, 8, or 9.	-
		Dynamics modeling	
6	Not available	No support: no BPMN concept fulfills requirement 6.	-
		Decision modeling	
6, 7, 8	Decision table	Decision-making based on inputs, static rule sets, and outputs (cf. DMN). No support: alternatives need to be compared to each other based on different criteria.	-

Support-Level (SL): + ⇒ full support, o ⇒ limited support, - ⇒ no support

BPMN lacks modeling support in the areas of process dynamics and decision-making related to communication. Connectivity issues with participants may lead to failing process executions (cf. req. 6). Available alternatives not explicitly modeled may not be used. For instance, there may be other participants in the proximity of a running process offering the required service of a missing participant that is part of the original process model. These participants may serve as an alternative to prevent failures.

Utilization of the *Decision Model and Notation (DMN)* to satisfy requirements 7 and 8 was considered but rejected: In DMN, a decision is taken using inputs and a set of rules structured by a decision table, resulting in an output [129]. However, deciding on alternatives for broken communication requires comparing different options with each other on the basis of defined criteria.

In conclusion, the equivalence check in Tables 3.1 and 3.2 identifies BPMN concepts for requirements 1, 2, 3, and 4. The check also demonstrates the need for an extension

to satisfy the requirements 5, 6, 7, 8, and 9 for communication, dynamics, and decision modeling.

3.3 Design of the Extension Concepts

The primary purpose of *rBPMN* is to introduce approaches satisfying the remaining requirements for resilient processes in unreliable communication environments. *rBPMN* defines so-called *resilience strategies for process modeling*, seeking to eliminate the negative consequences of connectivity issues on process operation. The following strategies are defined for the modeling of resilient processes:

Addition of alternatives If communication with a collaborative participant is vulnerable to connectivity issues, other participants may be added to the process model. The additional participants serve as alternatives for the initial participant, reducing the danger of failing process executions.

Exploitation of process dynamics At process runtime, dynamically appearing participants may serve as an alternative for unavailable participants. By integrating dynamics into process models, models may respect alternatives that have not been considered or even known of at design time.

Movable functionality In case of no connectivity, locally moved functionality may serve as a backup for unavailable participants. A process is enabled to continue its operation.

Resilience verification of process models By verifying the resilience of process models at design time, domain experts are enabled to identify and optimize vulnerable process segments before process runtime.

In conjunction with the determined requirements, the resilience strategies for process modeling guide the development of new modeling concepts. Subsequently, *rBPMN's* new modeling concepts and their graphical representations are illustrated. While this chapter introduces modeling-based resilience strategies, chapter 6 (p. 123ff.) complements the resilience strategies by introducing approaches for the process execution.

A key role regarding communication is taken by message flows, which describe the exchange of information between collaborative participants. However, BPMN message flows are missing the ability to state their demands in terms of technical aspects, such as the data size of the message to be transferred. Message flows are unable to indicate possible connectivity issues. Further on, a domain expert is unable to characterize message flows as alternatives to each other. Hence, *rBPMN* adds new message flow

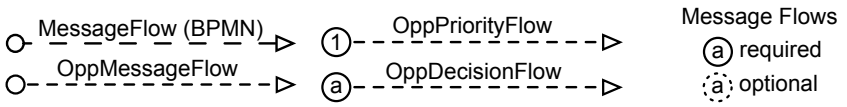

Figure 3.2: *rBPMN* message flows.

types to the BPMN modeling pallet. Figure 3.2 depicts the graphical representations of the extension concepts related to communication modeling.

Opportunistic Message Flows (abbreviated: *OppMessageFlows*, cf. Figure 3.2) describe possibly intermittent or broken communication segments. *OppMessageFlows* may be annotated with communication requirements and scenario-based connectivity descriptions to enable the verification of message flow resilience at process design time. OppMessageFlows may be used along with existing BPMN activities, participants, and traditional message flows.

Opportunistic Priority Flows (*OppPriorityFlows*) and *Opportunistic Decision Flows* (*OppDecisionFlows*) represent specializations of *OppMessageFlows*. They allow to define sets of alternatives in case of broken connectivity. With *OppPriorityFlows*, each message flow within a so-called alternatives group is labeled with a priority for decision-making. Priorities are represented by numbers, assigned to the modeled alternatives by domain experts at model design time. For instance, the *OppPriorityFlow* depicted in Figure 3.2 is assigned with priority 1, representing the highest-ranked alternative. This way, experts are provided with a tool to order alternatives against each other. The highest-prioritized *OppPriorityFlow* that is available at process runtime and satisfies the connectivity requirements is chosen.

A dynamically adapting modeling option for selecting alternatives is provided by *Opportunistic Decision Flows* (*OppDecisionFlows*). A character within the circle of an *OppDecisionFlow* labels its belonging to an alternatives group. For instance, the character *a* is specifying the alternatives group of the *OppDecisionFlow* depicted in Figure 3.2. In contrast to *OppPriorityFlows*, no fixed order of alternatives exists. Instead, decision-making is based on identifying and balancing characteristics of the alternatives at process runtime. According to the configuration provided by a domain expert, criteria such as resilience, accuracy, cost, and time of each alternative are compared to identify the best-suited available option. Experts may prioritize and combine criteria with each other. For instance, every resilient alternative may be evaluated using a weighted combination of the criteria accuracy, cost, and time. This allows to continuously optimize process operation within the given scenario opportunities. Chapters 4 and 5 (p. 65ff.) illustrate how graphs allow deciding on available alternatives (cf. Figure 1.1, p. 7). Besides, other decision-making tools such as WSM outlined in section 6.2.2 (p. 133ff.) can be used for decision-making.

With *required* and *optional*, two variants of the opportunistic message flow types (*OppMessageFlows, OppPriorityFlows, OppDescisionFlows*) exist. Using the required variant, one of the opportunistic message flows part of an alternatives group needs to be available for resilient operation. This is graphically indicated by a solid circle containing the alternatives group label (cf. Figure 3.2). As a second type variant, message flows may be optional in terms of resilient operation. This is illustrated by opportunistic message flows containing a dashed circle for the alternatives group label.

An essential part of *rBPMN's* resilience strategies is the provisioning of so-called movable functionality among participants. In the context of this thesis, functionality is defined as movable if it is capable of being transferred to another participant and executed there. Local execution does not necessarily provide the full set of features, it may represent a limited version of the original functionality. However, it is a strong tool to prevent failing process executions in the case of insufficient connectivity.

Offering movable functionality to other participants requires having a common understanding of what movable functionality is, how it is provided and how it can be used. The ontology in Figure 3.3 is used to extend the definition of functionality. The ellipses in the ontology describe concepts, which have relations (arrows) to other concepts. Key points at the *Metadata* and *Req.* concepts describe concrete instances represented by these concepts. The ontology leads to an extension of the functionality definition in terms of development, offering, and usage: *Functionality* is provided by *Services*, is described semantically by *Metadata*, includes *Input/Output Parameters*, and has a *Path-URL* based on the service's *Base-URL*. For identification, every service and functionality is labeled with a unique *ID* as part of its metadata. Finding suitable functionality is guided by a taxonomy, grouping services into *Categories*.

BPMN is missing concepts for providing and using movable functionality. Since functionality is part of collaborative participants and their tasks, a specialization of these concepts is required. Figure 3.4 illustrates the new task and participant types

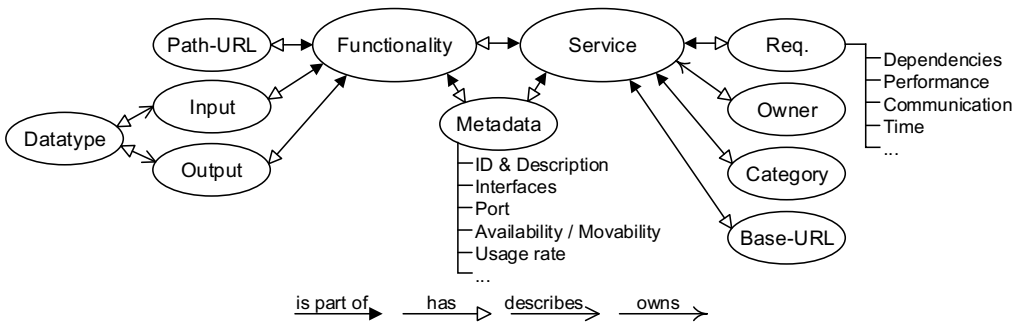

Figure 3.3: Ontology describing provisioning of functionality among participants.

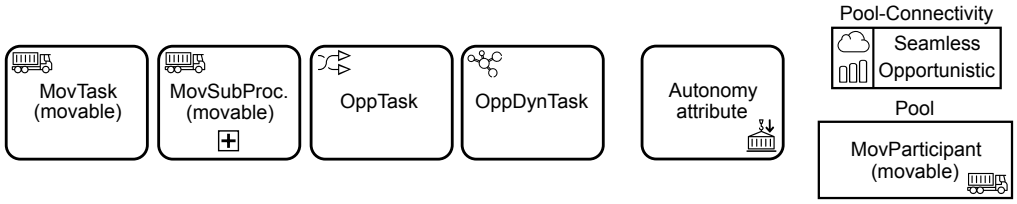

Figure 3.4: *rBPMN* tasks, attributes, and pools.

introduced by *rBPMN*. *Movable Tasks* (*MovTasks*), *Movable Sub-Processes* (*MovSub-Processes*), and *Movable Participants* (*MovParticipants*) offer movable functionality to other participants / system parts. In case of connectivity issues, the locally transferred functionality acts as a backup, allowing process operation to continue.

The required flexibility to execute locally moved functionality is provided by *Opportunistic Tasks* (*OppTasks*). Local functionality becomes part of the decision-making process as an additional alternative in the corresponding alternatives group. *Opportunistic Dynamic Tasks* (*OppDynTasks*) extend flexibility by the dynamic identification of suitable alternatives in the current scenario at process runtime. An *OppTask* and an *OppDynTask* are depicted in Figure 3.4.

Graphical attributes for *seamless* (cloud sign) and *opportunistic connectivity* (signal bar) of participants are defined (cf. Figure 3.4). The autonomy attribute of tasks allows to graphically indicate locally moved functionality as a backup for failing communication. All attributes do not affect process operation. Their purpose is to graphically point out characteristics of the modeling element.

Two examples illustrate the use of the new modeling concepts in real-world scenarios exposed to unreliable connectivity. Figure 3.5 depicts an *rBPMN* model of a heating process named *H*, heating up water for a water tank. To determine the required boil time, the current water temperature and the outside air temperature are identified using sensors. The process is placed in an unreliable WSN. The attributes for opportunistic connectivity in the upper left corners of the participants outline that connectivity with sensors may be interrupted or non-existing.

After the start of process *H*, the current water temperature is identified. Two temperature sensors are located within the water tank. *Water sensor 1* is the primary sensor located in the upper area of the tank while *Water sensor 2* represents a backup sensor which is located in the middle of the tank. Since connectivity is opportunistic, a domain expert uses *OppPriorityFlows* to integrate the two alternatives (cf. Figure 3.5). This way, the expert can define *Water sensor 1* as the primary alternative with the highest priority ($\Rightarrow$ priority 1) and *Water sensor 2* as the second alternative ($\Rightarrow$ priority 2).

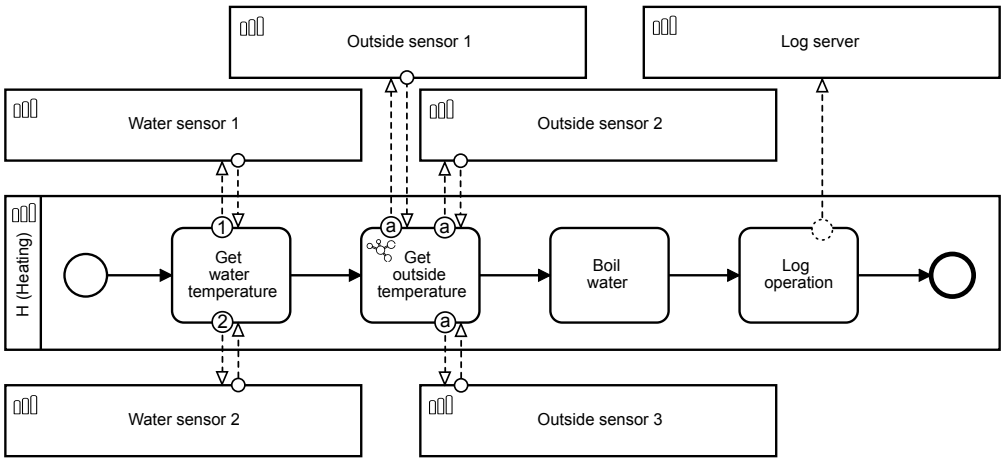

Figure 3.5: Heating process H part of a water heating system, located within a WSN (modeled in *rBPMN*).

Different air temperature sensors are located in the region of the heating system. Besides the three modeled outside sensors, additional sensors may appear and disappear dynamically. The domain expert uses *OppDecisionFlows* to label the three sensors as alternatives to each other. The alphabetic character *a* defines the three sensors as part of an *OppDecisionGroup*, where only one alternative has to be available for resilient operation (cf. Figure 3.5). As a decision criterion, the location of the sensor is chosen. This allows the process to select the outside sensor that is in the closest proximity of the heating system. Since other sensors may join the WSN, an *OppDynTask* is chosen to dynamically integrate sensors that have not been modeled. This way, the best-suited air temperature sensor may be chosen.

After identification of water and outside air temperature, the heating system heats up the water. Then, the heating process may log its operation by sending a message to a *Log server*. However, since this step is considered as not relevant for the resilient operation of the heating system, the *OppMessageFlow* is declared as optional.

The scenario of a heating system placed within a WSN illustrates the versatility of *rBPMN's* modeling concepts. The *OppDecisionFlows* in conjunction with the *OppDynTask* do not only guarantee optimal operation in terms of connectivity, but also regarding typical issues faced in WSNs. Sensors may be in sleep mode for energy-saving reasons, may break down, or be replaced by other sensors. These aspects mean no harm to process H, which continues to operate optimally under the given circumstances.

In a second example, movable functionality is used in a disaster scenario. The scenario depicted in Figure 3.6 consists of three rescue teams (Rescue participants) with additional supporters and a central *Crisis Management Group* (CMG). The *rescue teams* are instructed to drive into the disaster zone gathering and reporting information.

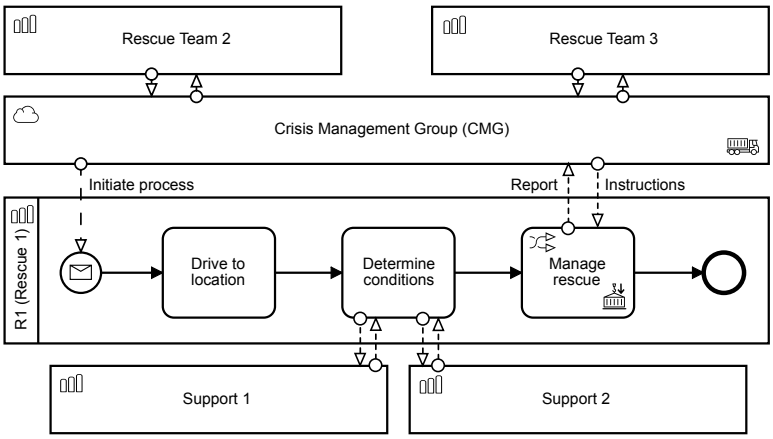

Figure 3.6: Rescue process *R1* as part of a disaster scenario (modeled in *rBPMN*).

The *CMG* collects information, analyzes risks, prioritizes actions, and coordinates the disaster relief by instructing the rescue teams.

The rescue process *R1* is initiated by a message flow, providing instructions about the location to be examined by the rescue team. This initializing message is modeled as a BPMN message flow, not as an *OppMessageFlow*. Since the rescue team is still in proximity of the *CMG*, connectivity is considered seamless. This changes when the rescue team arrives at the disaster zone. Communication of *R1* with *Support 1* and *Support 2* is modeled using *OppMessageFlows* since connectivity issues may occur when the rescue team spreads out across the disaster zone to determine conditions. Further on, reporting the conditions and receiving instructions from the *CMG* may be delayed and interrupted. Hence, the *CMG* allows moving functionality providing recommendations for the rescue management based on the determined conditions. The rescue team is enabled to rapidly organize and provide assistance as an autonomous unit.

The disaster scenario illustrates the combined use of BPMN message flows and *OppMessageFlows* in a common process model along with the local execution of functionality supporting the rescue management. Other real-world examples for movable functionality are provided as part of the agricultural slurry process, evaluated in chapter 7 (p. 139ff.).

3.4 Context Domain Model of the Extension

As illustrated in Figure 3.1, the definition of the CDME is the fourth step in developing a BPMN extension. The CDME allows describing the new modeling concepts of the extension domain and their relation to existing concepts of the BPMN metamodel in

an UML class model. Classes are identified as *existing* by using the stereotype *BPMN-Concept (BPMN Con.)* and as *new* by using the stereotype *ExtensionConcept (Ext. Con.)*. Restrictions of the BPMN extension mechanism can be ignored at this stage of development. The fifth development step in the following section will translate the CDME into an extended BPMN metamodel in compliance to the extension mechanism.

rBPMN's CDME is split into Figures 3.7 and 3.8 for better readability. In addition to the stereotype declarations, BPMN concepts are colored in gray and extension concepts are colored in white. BPMN concepts originate from the BPMN metamodel, being part of the BPMN specification [125]. The reader is referred to the BPMN specification for additional background information.

Figure 3.7 illustrates the communication- and decision-related concepts of *rBPMN*. The BPMN *BaseElement* acts as a root concept. BPMN concepts for *Participant, MessageFlow*, and *Activity* are included in the model since some extension concepts are inherited from these. All other extension concepts are directly inherited from BPMN's *BaseElement*. The corresponding inheritance arrows are omitted in Figures 3.7 and 3.8.

OppMessageFlows describe possible intermittent or broken communication parts and are derived from BPMN's traditional *MessageFlows*. This allows considering them as *MessageFlows* in environments not capable of interpreting *rBPMN* models. *OppMessageFlows* feature associations to numerous extension concepts related to communication and connectivity aspects.

MessageFlowProperties enable the description of the message to be transferred. Besides specification of the message size, an interval may be provided for recurring messages. The maximum delivery delay can be defined as part of the *QoSRequirements* concept. In particular for highly-frequent recurring messages, not every single message needs to arrive at the destination in many scenarios. Hence, a required delivery probability can be defined as QoS requirement. Grouping of messages is possible by allocating them to a *QoSPriorityClass*. Finally, connectivity properties such as a failure probability, a minimum and maximum bandwidth can be set for every *OppMessageFlow*. The properties describe the connectivity at the time of the message transfer. Since connectivity differs between scenarios and environments, multiple scenario-driven *ConnectivityProperties* can be defined for an *OppMessageFlow*.

Additional metadata for the handling of repetitions in processes can be added using the *RepetitionInfo* concept. Repetitions may occur for various reasons in a process and are often created by task and gateway combinations. Also, loop tasks and multi-instance activities result in repeating process segments (cf. [125] p. 36f.). Repetitions are challenging for the analysis of process resilience since the number of segment executions may vary. Metadata including the minimum, average, and maximum number

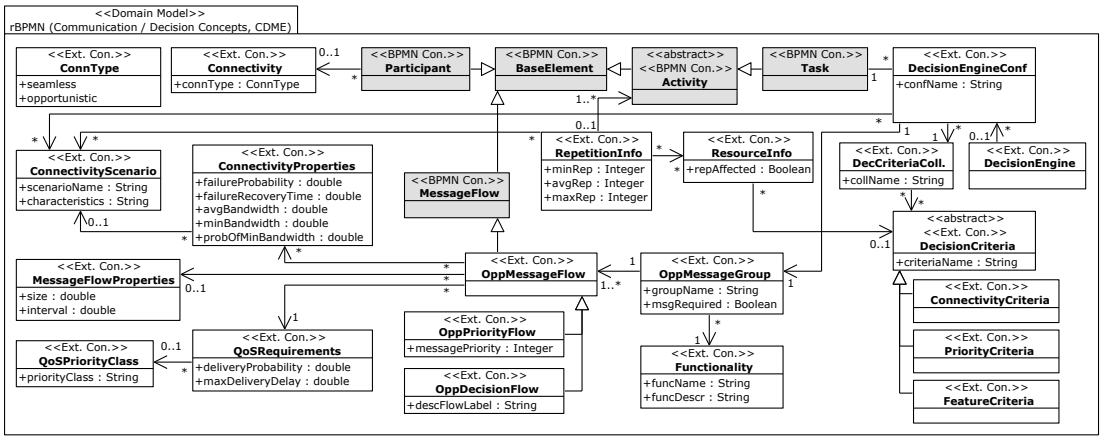

Figure 3.7: CDME of *rBPMN* including communication- and decision-related concepts.

Table 3.3: Extension concepts addressing communication and decision modeling.

Concept	**Semantics of communication modeling**
OppMessageFlow	Possibly intermittent or broken communication with other participants. May be used with existing BPMN concepts.
OppPriorityFlow	Opportunistic message flow with explicitly defined priority. A number within the message flow circle states the priority.
OppDecisionFlow	Opportunistic message flow with implicit, criteria-based decision-making for alternatives. An alphabetic character within the message flow circle states the *OppMessageGroup*.
OppMessageGroup	Group of *OppMessageFlows* that defines a set of alternatives.
MessageFlowProperty	Describes message properties (e.g., frequency, size, relevance).
QoSRequirements	Defines QoS requirements for a message flow.
QoSPriorityClass	Defines a QoS hierarchy, to be used by *QoSRequirements*.
RepetitionInfo	Provides metadata to address repeating process segments.
ResourceInfo	Specifies whether or not resources of a criterion are affected by repetitions (e.g., available connectivity, accuracy of task, cost of task).
Connectivity	Defines a type of connectivity (seamless, opportunistic) for a participant.
ConnectivityProperties	Describes connectivity at the time of a message flow.
ConnectivityScenario	Allows to group *ConnectivityProperties*, *RepetitionInfos*, and *DecisionEngineConfs* to different scenarios.
	Semantics of decision modeling
DecisionEngine	Chooses *OppMessageFlows* based on engine configuration.
DecisionEngineConf	Configures *DecisionEngine*, assigns collection of criteria.
DecCriteriaColl.	Collection of *DecisionCriteria* to be used as foundation for decisions.
DecisionCriteria	Criteria used by *DecisionEngine*. Available criteria: *ConnectivityDecision*, *PiorityDecision*, *FeatureDecision*.

of repetitions is beneficial and used as part of the resilience analysis of a process model (cf. chapter 4, p. 65ff.). Also, the effect of repetitions on resources such as available communication capacities can be specified by using the *ResourceInfo* concept. While in some scenarios additional time frames are available, other scenarios may have to share the communication resources calculated for a single execution of the repeating segment.

Descriptions for message properties, QoS requirements, and connectivity are used to verify the resilience of message flows at design time. This allows domain experts to verify and optimize resilience before actual process executions fail at runtime. Verification can be based on simple or detailed statistics of comparable processes. Alternatively, connectivity estimations may be used. Although the extension concepts provide a variety of options, not every detail is required for a resilience verification. The verification method is presented in section 3.6.

OppPriorityFlows and *OppDescisionFlows* represent specializations of *OppMessageFlows*. They are grouped as alternatives in *OppMessageGroups* and are configured as required or optional for resilient operation. Concepts for a decision engine, a decision engine configuration, and decision criteria allow the determination of the best-suited alternative based on connectivity, priorities, and features. For instance, a domain expert may define minimum connectivity characteristics for message flows and include criteria regarding the accuracy and cost of tasks addressed by the message flows. While the following chapters introduce comprehensive approaches for decision-making based on graphs, *rBPMN* is not requiring the use of a certain decision-making technique. For instance, decision-making based on WSM is illustrated as an alternative for appropriate scenarios in section 6.2 (p. 131ff.). Hence, the decision-making-related concepts of *rBPMN's* CDME are designed with flexibility in mind. The extension concepts for communication and decision-making aspects are listed in Table 3.3.

The collaboration-related concepts of *rBPMN* are depicted in Figure 3.8. A *Functionality* concept is used to ensure the consistency of movable functionality across the process model. Functionality may be described directly in the model. Alternatively, a reference to an external description is provided.

Movement of functionality is supported by extending BPMN's concepts for *Task*, *SubProcess*, and *Participant*. *MovTask, MovSubProcess*, and *MovParticipant* offer functionality to be used by other participants and system parts. The concept *FuncImplementation* describes a concrete implementation of movable functionality. This includes details about the required technology and performance as well as details about the functionality-image (e.g., format, size, link).

OppTasks allow to locally execute moved functionality and integrate it as an additional alternative. The concept *LocallyMovedFunc* is used to bind concrete implementations of functionality to *OppTasks*. This way, it is possible to add multiple imple-

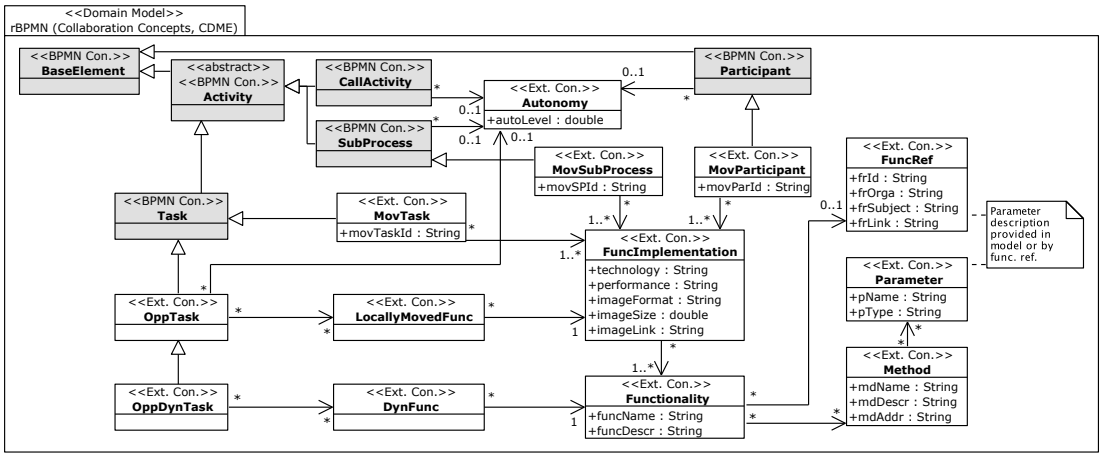

Figure 3.8: Continued CDME of *rBPMN* including collaboration-related concepts (modeled in UML).

Table 3.4: Extension concepts addressing collaboration modeling.

Concept	**Semantics of collaboration modeling**
Functionality	Defines and ensures consistency of functionality.
Method	Describes method of functionality in process model.
Parameter	Describes parameter of method in process model.
FuncRef	Describes functionality using an external link.
FuncImplementation	Describes a concrete implementation of functionality.
MovParticipant	Participant offering movable functionality.
MovTask	Task offering movable functionality.
MovSubProcess	Sub-process offering movable functionality.
OppTask	Task capable of executing locally moved functionality.
LocallyMovedFunc	Adds locally moved functionality to an *OppTask*.
OppDynTask	An *OppTask* capable of dynamically identifying alternatives not explicitly modeled at design time.
DynFunc	Adds dynamic functionality to an *OppDynTask*.
Autonomy	Defines autonomy level for tasks (e.g., four *OppMessageFlows*, three with local functionality ⇒ autonomy level of 75 percent).

mentations of the same functionality, offered by different participants. In contrast to *OppTasks*, *OppDynTasks* may be used to integrate dynamically appearing participants that have not explicitly been modeled at design time. The concept *DynFunc* binds functionality that may be discovered dynamically at runtime to *OppDynTasks*. All extension concepts for collaboration are elaborated in Table 3.4.

The design of *rBPMN's* CDME focuses on enabling a lightweight implementation without the need to implement the whole set of extension concepts. For instance, a subset of concepts referred to as *rBPMN-min* allows the verification of resilient process operation by using *OppMessageFlows* and related annotations for *MessageFlowProperties*, *QoSRequirements*, and scenario-driven *ConnectivityProperties*. Other scenarios may choose their own subset of concepts, depending on the scenarios' demands. Alternatively, the full set of concepts ($\Rightarrow$ *rBPMN-max*) may be employed.

Different technical opportunities are available for the implementation of concepts such as movable functionality and decision-making on alternatives. If useful, *rBPMN* may be extended or combined with existing extensions to support additional requirements and implementation strategies. Integration of BPMN concepts with low practical usage is avoided (e.g., message/item def. in BPMN runtime engines such as [25], cf. [76] for more details).

3.5 Metamodel of the Extension

An extension of the BPMN metamodel is needed to integrate the new concepts addressing the requirements for resilient process models. Using an MDA-based approach, the CDME is translated into the BPMN+X model (cf. step 5 in Figure 3.1). BPMN+X is realized as an UML profile and was introduced by [159], describing the extension in terms of the BPMN extension mechanism.

The BPMN+X UML profile is depicted in Figure 3.9. New stereotypes are introduced, based on the semantics and syntax of the BPMN extension mechanism. All concepts of an extension are grouped within an *ExtensionModel*, serving as an extension package. *BPMNEnum* describes an existing enumeration of the BPMN metamodel while *ExtensionEnum (Ext. Enum)* describes a new enumeration of the extension. Using *BPMNElement (BPMN Ele.)*, existing concepts of the BPMN metamodel may be represented. *ExtensionElement (Ext. Ele.)* describes a new concept added by the extension. With *ExtensionDefinition (Ext. Def.)*, groups of new concepts are created and named. All concepts of the group can be jointly added to existing BPMN concepts. The *ExtensionRelationship (Ext. Rel.)* is used to link an *ExtensionDefinition* to a *BPMNElement* for the purpose of extending the *BPMNElement*.

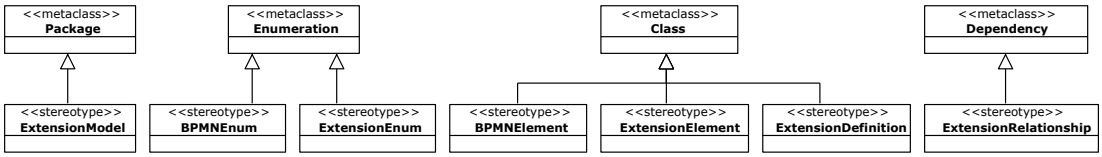

Figure 3.9: BPMN+X UML profile [159].

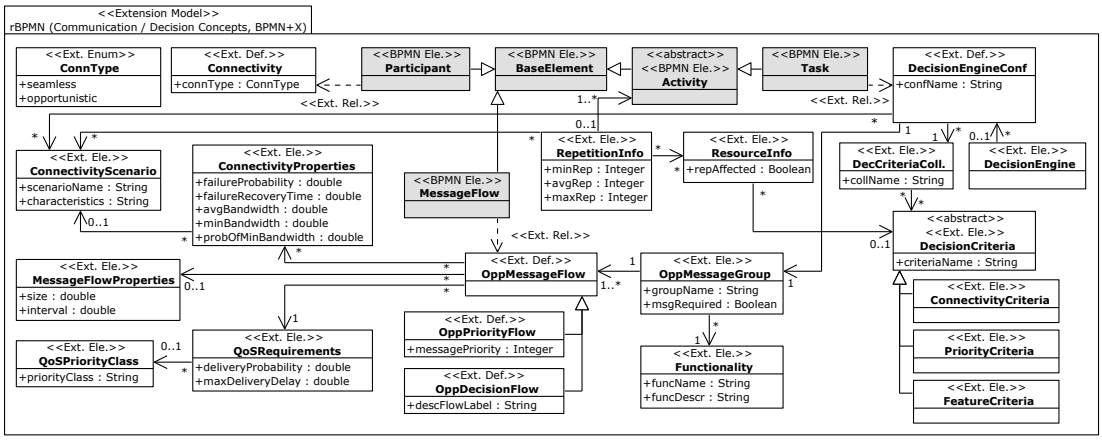

Figure 3.10: Metamodel of *rBPMN* (BPMN+X modeled in UML) including communication- and decision-related concepts.

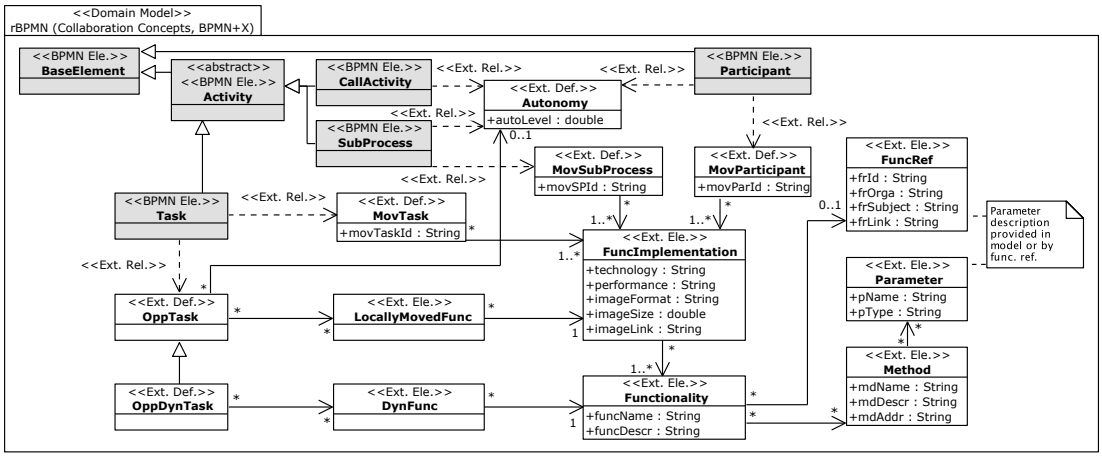

Figure 3.11: Continued metamodel of *rBPMN* (BPMN+X modeled in UML) including collaboration-related concepts.

Using the 15 transformation rules of [159], the CDME is transformed into the BPMN+X model depicted in Figures 3.10 and 3.11. *BPMNConcepts* and *ExtensionConcepts* do no longer exist since they have been transformed into *BPMNElements*, *ExtensionDefinitions*, *ExtensionElements*, and *ExtensionEnums*. In Figures 3.10 and 3.11, existing *BPMNElements* are colored in gray while all extension-related classes are colored in white. Generalizations and associations with *BPMNElements* have been transformed to *ExtensionRelationships*. Since the transformation method is not modified but simply applied in this thesis, a detailed explanation of the method is omitted. Information about the transformation rules and their application is provided by [159].

In the final step of the extension development procedure illustrated in Figure 3.1, the BPMN XML schema extension definition document is created. This XML schema represents an exchange format for extensions to be used by BPMN tools. While this may allow to graphically inspect *rBPMN* models in existing BPMN tools, it is insufficient for the execution of *rBPMN* models. Chapters 6 and 7 (p. 123ff.) elaborate in detail on the additional efforts necessary to execute *rBPMN* process models.

The creation of the XML schema is guided by model transformations. At first, the BPMN+X metamodel is transformed into an XML schema extension definition model. Afterward, a second transformation is applied to create the XML schema. Along with the required transformation rules, [159] provides a tool for automating the two model transformations.

For the reasons presented, the final step of creating the XML schema is omitted in this thesis.

3.6 Resilience Verification of Message Flows

Using the communication-related concepts elaborated in the previous section 3.4, the resilience of message flows can be verified by *i)* calculations based on connectivity characteristics or by *ii)* connectivity probabilities. Depending on the knowledge about a process, the available process data, and the objective of the resilience analysis, either one or the other approach may be appropriate for a scenario. Either way, data used to analyze message flow resilience can be based on *a)* connectivity estimations or on *b)* connectivity statistics gathered in previous or comparable process executions. Subsequently, the resilience verification of message flows using connectivity characteristics and connectivity probabilities is illustrated.

3.6.1 Connectivity Characteristics

The resilience verification based on connectivity characteristics calculates the required amount of time to transfer data between participants. By comparing the calculated data transfer time with the allowed message delivery delay defined in the QoS requirements, it is verified whether or not the transfer finishes in time. The difference between calculated and allowed delay time also indicates impacts of possible deviations in expected and actual connectivity at runtime. For instance, domain experts should enhance the resilience of a message flow that barely fits into the required delivery time. In contrast, the calculated statement is unlikely to change to non-resilient if there is time to transfer the message multiple times.

The first step in the resilience verification of a message flow is to calculate the required number of data frames N_f. Therefore, the message size M_s is divided by the frame payload size F_{pl} as illustrated in Equation 3.1.

$$N_f = \left\lceil \frac{M_s}{F_{pl}} \right\rceil \tag{3.1}$$

With the number of frames on hand, the time it takes to transfer the required data frames can be calculated. Equation 3.2 provides a formula to calculate a basic time T_b by including the minimum bandwidth BW_{min} in conjunction with the message size and the frame header size F_h.

$$T_b = \frac{M_s + N_f * F_h}{BW_{min}} \tag{3.2}$$

Alternatively, an advanced time T_{adv} may be calculated by combining minimum bandwidth BW_{min}, average bandwidth BW_{avg}, and their probabilities, resulting in a common bandwidth BW in Equation 3.3.

$$BW = BW_{min} * P_{BWmin} + BW_{avg} * (1 - P_{BWmin}) \tag{3.3}$$

Further on, the effects of transfer failures are integrated into T_{adv}. For this purpose, a function $F_{(P_f,y)}$ is defined in Equation 3.4, using the packet transfer failure probability P_f and a random number $y \in [0, 1]$ generated by a random number generator with equal distribution.

$$F_{(P_f,y)} = \begin{cases} 1 & \text{if } P_f < y, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

Following, the time required for the transfer is calculated and combined with a summarized time frame for failure recovery. Depending on P_f, a failure recovery time T_f is added multiple times for a data transfer. Subsequently, Equation 3.5 illustrates the calculation of T_{adv}.

$$T_{adv} = \frac{M_s + N_f * F_h}{BW} + \sum_{i=1}^{N_f} F_{(P_f, y_i)} * T_f \tag{3.5}$$

Comparing the maximum allowed delivery delay for a message flow T_d with the actual time required for the transmission $T_{b/adv}$ reveals whether or not a message flow is *i)* resilient ($T_d \geq T_{b/adv}$) or is *ii)* non-resilient ($T_d < T_{b/adv}$).

For repeating message flows, the message flow interval T_i can be divided by $T_{b/adv}$ to get the number of messages N_m able to be transferred within the interval (Equation 3.6). The resilience of repeating message flows depends on the required delivery probability P_d. Process operation is resilient for $N_m \geq P_d$ and non-resilient for $N_m < P_d$. For instance, if periodic status messages of a process require $P_d \geq 0.5$, up to 50 percent of the status messages may be lost and communication is still considered as resilient.

$$N_m = \frac{T_i}{T_{b/adv}} \tag{3.6}$$

Domain experts may have difficulties estimating the required parameters for the resilience calculations of message flows. Provisioning of representative bandwidth values for poor and average connectivity in the application domain helps to improve calculations. Guidelines for typical protocol stacks with their frame/packet headers and payload sizes can be provided. As an example, Figure 3.12 illustrates the encapsulation of application data (layer 7) into TCP/IP (layers 4 and 3), an 802.11 MAC (Media Access Control) frame with a tailing Frame Check Sequence (FCS, layer 2), and into the Physical Layer Convergence Protocol (PLCP) with preamble and header (layer 1) [149] [148] [74]. Tools for monitoring and evaluating communication characteristics at process runtime may identify more precise values for concrete scenarios.

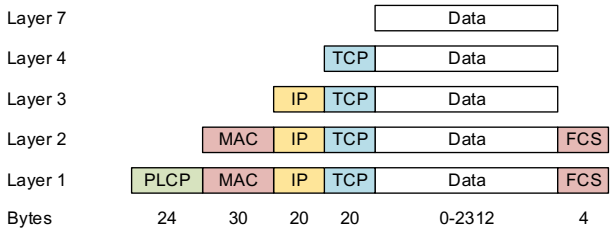

Figure 3.12: A data frame including protocol headers/tailers for 802.11 (WiFi), IP, and TCP.

Verifying the resilience of a message flow is illustrated in an example: Collaborative participants exchange information wirelessly using a data transfer. A WiFi network is used in combination with TCP/IP. The size of the message to be transferred is defined as $M_s = 1\ MByte$, the minimum bandwidth as $BW_{min} = 1\ Mbps$, and the maximum delivery delay as $T_d = 10\ s$. According to Figure 3.12, the frame payload size is $F_{pl} \leq 2312\ Byte$ while the frame header has a size of $F_h = 98\ Byte$. Using Equation 3.7, the number of frames is identified as $N_f = 454$.

$$N_f = \left\lceil \frac{1\ MByte}{2312\ Byte} \right\rceil = 454 \tag{3.7}$$

Following, Equation 3.8 determines the time required for the data transfer with $T_b = 8.74\ s$. Since the required time for the data transfer is smaller than the maximum delivery delay ($8.74\ s < 10\ s$), the message flow is considered to be resilient.

$$T_b = \frac{1\ MByte + 454 * 98\ Byte}{1\ Mbps} = 8.74\ s \tag{3.8}$$

The equations for the resilience verification of message flows abstract some technical details of communication to avoid overextending domain experts. For instance, the equations do not consider collisions in shared WiFi frequency bands, bit error rates of data transfers and window sizes of protocols such as TCP. If required, the equations may be adapted to the scenario's needs.

3.6.2 Connectivity Probabilities

A different perspective is provided by indicating the resilience of message flows with probabilities. This often results in a more abstract view for domain experts on resilience, less bound to technical parameters of data transfers. Scenarios that gathered simple statistics about the success and failure of message flows may benefit. Besides, probabilities may be helpful in scenarios where only rough estimations about the message flow resilience can be provided.

No general applicable formulas for the resilience verification of message flows can be provided when working with connectivity probabilities: Depending on the scenario, the probabilities may be extracted from statistics or assigned based on estimations. When gathered from statistics of previous process executions, the resilience probability of a message flow is represented by the value of successful versus unsuccessful message flows that have taken place.

Summary

The process model is a major aspect of operating processes in unreliable communication environments. Designing resilient operation requires modeling elements that allow for comprehensive process adaptations during runtime. Models need mechanisms to include and decide on alternatives for possibly failing communication. A model resilience verification mechanism takes an important role in designing processes by allowing early identification and optimization of vulnerable process parts.

rBPMN introduces new strategies to support the modeling of resilient processes and the verification of resilient operation at design time. In case of intermittent or broken connectivity, *rBPMN* allows *i)* to add alternatives for failing message flows, *ii)* to find appropriate alternatives dynamically at runtime, and *iii)* to move functionality among participants as local backups. Domain experts can specify optimal process operation explicitly by assigning priorities to alternatives or implicitly by choosing alternatives dynamically based on specified characteristics. The resilience of message flows can be identified using the provided calculations for connectivity characteristics or by using connectivity probabilities. Either connectivity examinations or gathered statistics of previous process executions serve as foundations for the resilience verification.

Approaches for the realization of *rBPMN's* resilience strategies and the resilience verification are illustrated in the following chapters. The versatility between the extended metamodel (BPMN+X), its concepts, and the realizing implementation techniques is an important aspect of the lightweight design of *rBPMN*: While the presented approaches are beneficial for many application domains, only a subset of the approaches may be used, depending on the requirements of the scenario. The lightweight design allows to apply *rBPMN* in many different use cases and application domains. Also, different implementation techniques (e.g., for the realization of movable functionality or decision-making) may be used, if desired.

CHAPTER 4

GRAPH-BASED RESILIENCE ANALYSIS

Verifying the resilience of a process model at design time allows domain experts to identify and optimize imperfections of the model avoiding process failures at runtime. Most processes include different paths to traverse from start to end, also referred to as process configurations. While chapter 3 (p. 41ff.) illustrates how to calculate the resilience of individual message flows, a mechanism to analyze the resilience of process paths is missing. Multiple statements about the resilience of a process may exist, related to its different paths. For instance, if a process includes multiple endpoints, it may be resilient and non-resilient at the same time. This is the case if resilience is only identified for some of the endpoints. Due to the possible varieties, a precise definition of resilient process operation is introduced subsequently.

This chapter presents a graph-based approach to examine the resilience of process models. The approach allows to compare and rank the different process configurations against each other. A variety of resilience metrics is introduced, allowing the selection of relevant metrics for different application domains and scenarios. A rule set for the translation of process models to resilience graphs is illustrated in detail. The resulting DAGs serve as a foundation for the graph analysis. Different types of graph algorithms are evaluated to identify resilient process configurations. Since the problem of finding resilient process paths differs from the well-known SPF problem, adjustments have to be made if these algorithms shall be used. Also, a novel approach of a graph algorithm combining different types of communication technologies to optimize resilience is presented.

4.1 Resilience Metrics

The main outcome of a resilience analysis in regards to unreliable communication environments is whether or not a process is able to operate without being interrupted or terminated by connectivity issues. Since the meaning of resilient operation may vary depending on the scenario and involved domain experts, a definition of resilient process operation is provided in the following subsection.

The key message of resilient operation is accompanied by a diverse set of values describing different aspects of process behavior. They may provide indications why a process is not capable of resilient operation and what can be done for its optimization. They allow comparing and ranking process configurations against each other. Multiple metrics measuring resilience properties are introduced subsequently. A distinction is made between resilience calculations based on connectivity characteristics and connectivity probabilities.

Resilient Process Operation

The calculations introduced by *rBPMN* in section 3.6 (p. 59ff.) allow stating if message flows can operate within the required QoS (⇒ resilient) or if the QoS requirements are not achievable (⇒ non-resilient). In contrast to the resilience of message flows, a statement about the resilient operation of a process is diverse. Many processes include numerous message flows, being part of different paths from the process' start to its end. A process can have multiple endings or terminate in an error state.

A definition of *resilient process operation* is provided by using process example *Ex0* in Figure 4.1. The example contains all relevant aspects required to provide a definition. *Ex0* features three different tasks, each of them communicating with other participants using message flows. For reasons of comprehensibility, the process has been divided into the two parts *a)* and *b)*. Hereafter, abbreviations in the form of *Ex0-a)* are used to refer to parts of process examples [i.e., *Ex0-a)* for part *a)* of *Ex0*].

Considering the first part *Ex0-a)* of the example, the process may include *i) no, ii) one/many,* or *iii) only* resilient process paths. In the case of *i)*, the process is defined as *non-resilient*. For the latter two cases, the process is defined as *resilient* since at least one non-interrupted path to the process' end is available. To point out the special characteristics of case *iii)*, the process is defined as *all-paths resilient*.

A second endpoint is added to the example in Figure 4.1 by *Ex0-b)*. The process is still considered resilient if only one endpoint can be reached without interruption. However, a domain expert may expect the process to end at the other endpoint and be misled by the term *resilient process*. Hence, the definition of an *all-ends resilient process* allows to easily distinguish a process that is resilient for all endpoints against

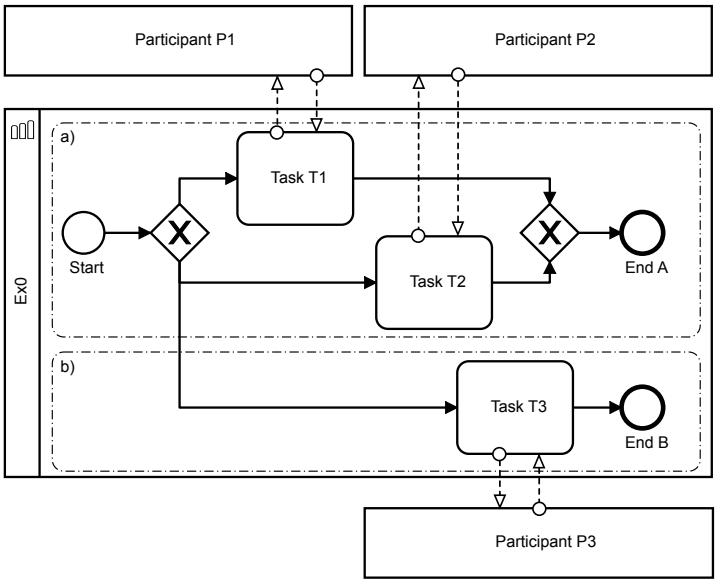

Figure 4.1: Process example *Ex0* (modeled in *rBPMN*).

Table 4.1: Definition of resilient process operation.

Process characteristics	**Process operation is ...**
No uninterrupted path	non-resilient
One/many uninterrupted path(s)	resilient
Only uninterrupted paths	all-paths resilient
One/many uninterrupted path(s) for every endpoint	all-ends resilient

a process that is only resilient for one or many endpoints. Table 4.1 summarizes the statements about resilient operation.

Having the resilience of processes and process paths defined, the question arises if one path may be more resilient than another. Since calculations determine if and how well the QoS requirements of message flows can be met, it is possible to rank message flows against each other. As illustrated subsequently, the same applies to different paths of a process.

Metrics for Connectivity Characteristics

rBPMN introduces an approach based on connectivity characteristics to calculate the resilience of message flows at design time. Parameters such as message size, protocol overhead, available bandwidth and failure probability are part of the calculation (cf. section 3.6.1, p. 60ff.). The approach validates whether or not a message flow is resilient. The resulting resilience value $R_e \in \mathbb{R} | R_e \geq 0$ of the calculation is used as a

weight for the corresponding edge in the resilience graph. R_e basically describes how often a message can be sent in an available time period (cf. Equation 4.1).

$$\begin{aligned} R_e < 1: &\quad \text{The message can be sent only partly.} \\ R_e = 1: &\quad \text{The message can be sent exactly once.} \\ R_e > 1: &\quad \text{The message can be sent more than once.} \end{aligned} \tag{4.1}$$

Summarizing the message flows that are part of a process path results in the total resilience of the path R_t. The resilience level of a path R_l represents its minimum path edge. Other metrics such as the highest, average, and median resilience of a path as well as the range of a path describe additional characteristics. The path difference R_d represents the summarized difference of the non-resilient edges of a path (u) to a resilient edge weight (r). Using the metrics, it is not only possible to find resilient process paths but to rank the resilience of different paths. Identified process paths may be compared by the metrics of Table 4.2.

Applying the metrics to sub-parts of a process path may be beneficial for stating and comparing the resilience characteristics of collaborative participants or specific process segments. For instance, $P1\,R_l$ specifies the resilience level R_l of participant *P1* in Figure 4.1.

Calculations are based on estimated parameters or statistical investigations and may differ from real-world connectivity. Raising the required resilience level R_l allows to include a *connectivity safety margin* for resilient process paths. For instance, a

Table 4.2: Resilience metrics.

Semantic	**Symbol & Formula**	**P**
Number of weighted path edges	$n \in \mathbb{N}$	✓
Resilience of a path edge	$R_e \in \mathbb{R}$	
(Total) Resilience of a path	$R_t = \sum_{i=1}^{n} R_{e_i}$	
Resilience level of a path	$R_l = min(R_{e_1}, ..., R_{e_n})$	✓
Highest resilience in a path	$R_h = max(R_{e_1}, ..., R_{e_n})$	✓
Average resilience of a path	$R_a = R_t/n$	✓
Median resilience of a path	$R_m = median(R_{e_1}, ..., R_{e_n})$	✓
Range of a path	$R_r = R_h - R_l$	✓
Difference of a path	$R_d = \sum_{j=1}^{u} \lvert r - R_{e_j} \rvert$ with u non-resilient edges and r as the scenario resilience value	

Declaration: ✓⇒ applicable for connectivity probabilities (P)

Table 4.3: Resilience metrics exclusive to connectivity probabilities.

Semantic	**Symbol & Formula**
Resilience probability of a path edge	$P_e \in \mathbb{R} \mid 0 \leq P_e \leq 1$
Resilience probability of a path	$P_p = \prod_{i=1}^{n} P_{e_i}$
Boolean resilience probability of a path	$P_b = 0 \quad (\forall\, P_p < 1)$ $P_b = 1 \quad (\forall\, P_p = 1)$

scenario may define a minimum $R_l = 2.0$ as a requirement, although the path operates resiliently with $R_l \geq 1.0$ ($\Rightarrow$ connectivity safety margin of $2.0 - 1.0 = 1.0$). A path with $R_l = 1.5$ would be rated as non-resilient since it is not able to meet the upper bound of 2.0 defined by the connectivity safety margin.

It is important to select appropriate metrics to verify process resilience. Also, it is often useful to consider multiple metrics during the verification, providing background information about the resilience statement. Since the metrics are applied to resilience graphs, the graph creation is elaborated first. Examples for the use of metrics follow in section 4.3.

Metrics for Connectivity Probabilities

Connectivity probabilities are defined as $P \in \mathbb{R} \mid 0 \leq P \leq 1$. Many of the metrics described in Table 4.2 are also applicable to probabilities. However, the resilience probability of a path P_p is defined as the product of the path edges P_e. The metric states the probability of resilient process operation for the chosen path. A boolean indication for the path resilience is defined as P_b, allowing easy identification of a resilient or non-resilient path. Table 4.3 summarizes the specific metrics for probabilities, examples of their use are provided in section 4.3.

4.2 Process-to-Graph Translation

Finding resilient process paths using graph algorithms requires the translation of the model into a graph first. This section introduces an approach to translate BPMN and *rBPMN* models to DAGs with respect to communication resilience. The translation rule set defined by the approach describes the fundamental paradigms of how process sequences are translated. After translation, the graph may be simplified and analyzed. Figure 4.2 visualizes the process steps of a resilience analysis.

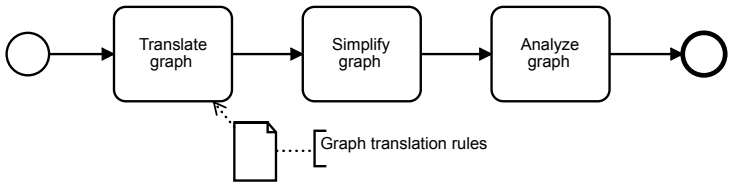

Figure 4.2: Process steps of a resilience analysis (modeled in BPMN).

4.2.1 Creation of the Resilience Graph

This thesis introduces the concept of *resilience graphs*, which are defined as DAGs, representing communication aspects. A resilience graph is the foundation of a resilience analysis. Subsequently, the translation of process models to resilience graphs is illustrated on the basis of two examples. Further on, a list of universally applicable translation rules for the process-to-graph translation is presented.

Figure 4.3 depicts a simple process example *Ex1*, containing exclusive gateways and a participant *P1*. As indicated by the *OppMessageFlows*, communication of task *T1* with *P1* may interfere. In contrast, the path segment including *T2* requires no communication and is accordingly resilient. Only one of the two paths including *T1* or *T2* is chosen depending on the parameters / process variables instructing the exclusive gateway with its decision.

The related resilience graph $G_{Ex1} = (V, E)$ includes a set of vertices V representing BPMN activities/participants and a set of edges E depicting sequence flows and

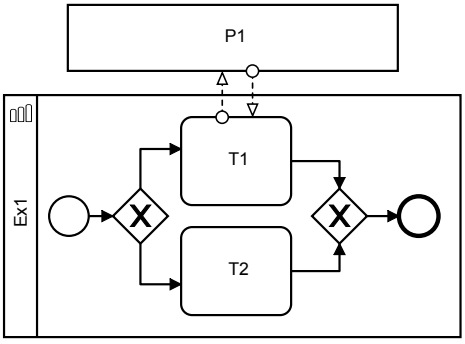

Figure 4.3: Process example *Ex1* (*rBPMN*).

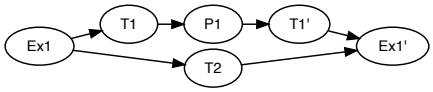

Figure 4.4: Resilience graph of *Ex1*.

message flows. As shown in Figure 4.4, G_{Ex1} has a starting vertex *Ex1* and an ending vertex *Ex1'*. Accordingly, communication of *T1* with *P1* is arranged by *T1* $\rightarrow$ *P1* $\rightarrow$ *T1'*. A second graph path represents the use of *T2* instead of *T1*.

A second process example *Ex2* is illustrated in Figure 4.5, featuring exclusive and parallel gateways. Communication with all participants is unreliable. The associated resilience graph in Figure 4.6 has been created using the translation rules listed in Tables 4.4 and 4.5. The translation for this example is explained in detail subsequently.

In the first process part *Ex2-a)* in Figure 4.5], the paths of *T1* and *T2* are separated by an exclusive gateway. Since only one of the two paths is chosen, the graph reflects this by adding separate paths for *T1* and *T2* and merging them afterward (*Ex2* $\rightarrow$ *G* in Figure 4.6).

At *T2*, communication with the participants *P2a* and *P2b* is realized by *OppDecisionFlows* belonging to the same *OppMessageGroup*, labeled with the character *a*. Either communication with *P2a* or *P2b* has to work for resilient operation of this path segment. Hence, *T2* connects *P2a* and *P2b* by separate paths, resulting in three path options for resilient operation in *Ex2-a)*. The vertex *G* is introduced as a glue vertex since there is no BPMN activity element merging the different process paths.

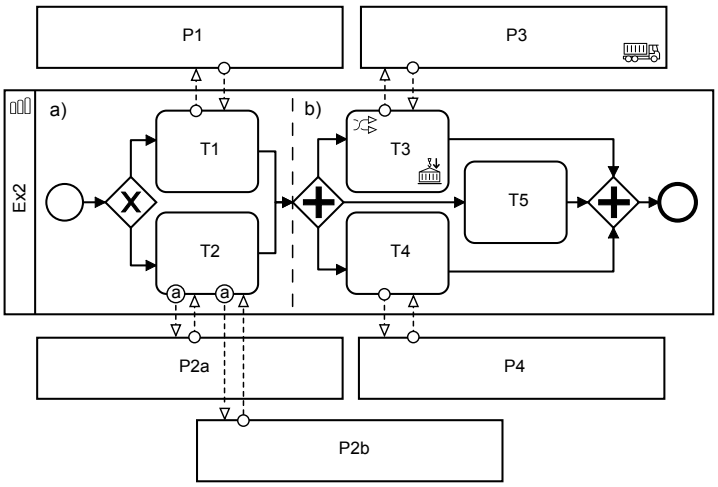

Figure 4.5: Process example *Ex2* (*rBPMN*).

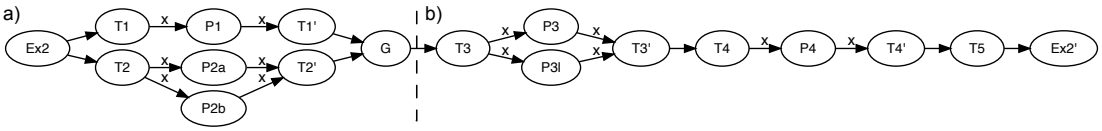

Figure 4.6: Resilience graph of *Ex2*.

The second process part *Ex2-b)* executes *T3, T4* and *T5* in parallel. *T3* and *T4* are influenced by unreliable communication, which is indicated by the *OppMessageFlows* to *P3* and *P4*. *T5* includes no communication and is always resilient. All three tasks need to be executed for a resilient process due to the parallel gateway. To reflect this in the graph, the path of *T3* is extended by the paths of *T4* and *T5*.

T3 needs to call a functionality offered by *P3*. This can be done by *i)* calling *P3* and receiving the desired result or by *ii)* moving (limited) functionality from *P3* to *T3* and execute it locally (*P3l*). The resilience graph integrates this by two separate paths between *T3* and *T3'*. Communication of *T4* with *P4* may interrupt or break. Since there are no decisions, its representation in the graph is a single path $T4 \rightarrow P4 \rightarrow T4'$. Lastly, T5 is resilient since it is not involved in any communication. By extending the path of *T3 / T4* with a vertex for *T5*, the resilience graph of Figure 4.6 is complete. The next subsection explains the x-labeled graph edges and why the inclusion of *T5* is dispensable in this scenario.

Tool support for an automated translation of BPMN processes to resilience graphs is conceivable. The translation may be automated following the rules of Tables 4.4 and 4.5. The tables list process elements and state if the translated graph segment includes separated paths, an extended path or if a different graph representation is necessary. As denoted, some elements may require process context information to clarify the domain expert's intention for the modeled process segment. The semantics of these elements are not explicit regarding communication aspects. For this reason, *rBPMN's* metamodel includes concepts to represent such information. For instance, the *RepetitionInfo* concept allows to specify metadata describing characteristics of repetitions. Alternatively, BPMN text annotations or other metadata fields may be used.

A proof of completeness regarding the translation of BPMN elements using the rules of Tables 4.4 and 4.5 is not part of this thesis. However, all collaboration-related elements of the BPMN relevant to resilient operation are addressed (cf. [125] p. 26-39), and also the translation of combinations of elements (e.g., merging exclusive gateways, various kinds of loops) is elaborated.

Table 4.4: Rules for the translation of BPMN process elements to graph segments.

Process element	Graph segment		Explanation
BPMN gateways (GWs)			
Exclusive GW, Event-based GW	Separated paths		Only one of the GW options is chosen and needs to be part of the corresponding process path.
Parallel GW	Extended path		Resilience depends on all GW options.
Inclusive GW, Parallel event-based GW	Separated and extended paths		One or more GW options can be chosen at the same time. Every possible GW option combination needs to be reflected in the graph.
Complex GW	Separated paths and / or extended path(s)		No general rule can be provided, depends on concrete GW options. Rarely used in practice, not supported by many BPMN runtime engines. Often replaced by other GW types.
BPMN path merging by exclusive GW			
Splitting GW: parallel, inclusive, parallel event-based, complex	Separated and extended paths	*	Splitting GW and merging exclusive GW result in multiple executions of the merged process segment. Calculation of repetition-based edge weights required if communication resources shall be shared.
BPMN flows / events			
Conditional sequence flow			Different way to model GW options. GW translation rules are applied.
Message flow	Extended path		Resilience depends on the success of the *message flow* (since BPMN has no *optional message flow* element).
Event			If relevant for resilience, the *event* is initiated by a *message flow*. *Message flow* translation rules are applied.
Interrupting event / signal	Separated paths		*Event / signal* occurrence modifies process path. Resilience depends on the alternative path initiated by the *event / signal*.
BPMN activities			
Sub-process, Call activity, Transaction	Extended path		Resilience depends on activities within the *sub-process / call activity / transaction*. Graph may be extended by a subgraph.

Declaration: * provisioning of process context information eliminates semantic gap

Table 4.5: Rules for the translation of BPMN and *rBPMN* process elements to graph segments.

Process element	Graph segment	Explanation
BPMN participants		
Pool Collapsed pool		Basically represents a (summarized) subgraph. Initiated and concluded by *message flows*. Translation rules of *message flows* are applied.
BPMN loops / multi-instance activities		
loop parallel sequential	a) Separated paths *	a) Separate paths for every number of executions. Calculation of repetition-based edge weights required if communication resources shall be shared.
	b) Extended path	b) If communication resources are not shared / if resilience is not affected.
***rBPMN* flows**		
OppMessageFlow, OppPriorityFlow, OppDecisionFlow	a) Extended path	a) If *opportunistic message flow* is declared as *required*. Caution: respect *OppMessageGroup* rules.
	b) Remove element	b) If *opportunistic message flow* is declared as *optional* (⇒ resilience is not affected by this *opportunistic message flow*).
OppMessageGroup	Separated paths	*OppMessageGroups* define sets of alternative flows. Hence, every flow within the set results in a separate path in the resilience graph.
Multiple OppMessage-Groups	Extended path	Multiple *OppMessageGroups* separate alternatives for different concerns, resulting in an extended path.
***rBPMN* activities / participants**		
OppTask (moved functionality)	a) Separated paths	a) Functionality is an alternative for an existing *OppMessageGroup*.
	b) Extended path	b) Functionality creates its own *OppMessageGroup*.
OppDynTask (dynamic participants)	Separated paths	*OppDynTasks* allow to integrate dynamically appearing participants as alternatives for existing *OppMessageGroups*.
MovTask, MovSubProcess, MovParticipant	Separated paths	A path for executing functionality on a remote participant. Another path for local functionality execution (⇒ *OppTask* rule).

Declaration: * provisioning of process context information eliminates semantic gap

4.2.2 Simplification of the Resilience Graph

The resilience graph in Figure 4.6 relates to the *rBPMN* process illustrated in Figure 4.5. However, only a subset of edges and vertices included in the graph is exposed to unreliable communication and its consequences for resilience. The affected edges have been labeled with an x, representing process parts where unreliable communication or moving of functionality occurs. Figure 4.7 illustrates a simplified version of the resilience graph shown in Figure 4.6. Unlabeled edges have been removed, vertices not connected to an incoming or an outgoing labeled edge have been removed or merged. For instance, the vertices *T1*, *T1'*, *T2*, *T2'*, and *G* add no benefits to the graph with respect to communication resilience. Hence, they have been removed. The outcome is a compact DAG, ready to be assigned with edge weights and to serve as a foundation for the graph analysis.

In general, process elements affected by resilience are of special interest for the graph translation. In the area of unreliable communication environments, this requires attention when translating:

- possibly failing message flows (*BPMN: MessageFlows / rBPMN: OppMessageFlows, OppPriorityFlows, OppDecisionFlows*),
- movable activities (*MovTasks, MovSubProcesses, MovParticipants* of *rBPMN*),
- locally executable functionality (*OppTasks* of *rBPMN*) and
- dynamic alternatives (*OppDynTasks* of *rBPMN*).

However, it is important to still include all possible process paths in the graph after simplification. If a path segment between gateways is resilient due to avoidance of communication, it may be reduced to a single vertex but still needs to be part of the graph. For instance, it is not allowed to remove *T2* from the graph of *Ex1* in Figure 4.4. Otherwise, a resilience analysis would not be able to identify the resilient path using *T2* anymore. *Ex1* would be identified as non-resilient although it includes a resilient path.

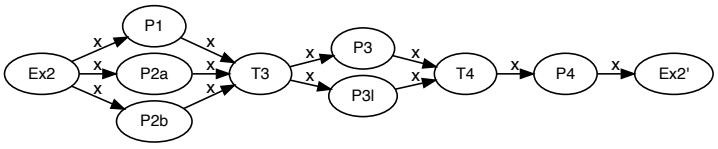

Figure 4.7: Simplified resilience graph of *Ex2*.

4.2.3 Further Translation Elaborations

Not all translation rules listed in Tables 4.4 and 4.5 have been applied in the translation of the process examples *Ex1* and *Ex2*. This section provides concluding examples for significant translation aspects based on (simplified) resilience graphs.

Loops and Multi-Instance Activities

A resilience graph is defined as a directed, but acyclic graph. Many business processes include repeating segments such as loops and multi-instance activities. A mapping eliminating the cycles of these path segments is required for graph algorithms analyzing process resilience.

In BPMN, a loop is often the result of a process flow including activities and gateways ([125] p. 36f.). Process example *Ex3-a)* in Figure 4.8 illustrates a loop created by task *T1* and the following exclusive gateway. The gateway repeats *T1* if instructed to do so by the condition of a process variable. Otherwise, execution continues with the remaining parts of *Ex3-b)* and *Ex3-c)*. The variable controlling the number of *T1*-repetitions may be affected by *i)* the running process itself or by *ii)* external processes or systems. Reasons for repeating *T1* can be diverse: For instance, *T1* may improve an approximation with every iteration. The number of repetitions may depend on measurements until reaching a certain level. Besides, repetitions may be defined by the number of objects in a list, by reading values of other system parts, by following time constraints or repetitions are simply predefined. Basically, three types of process repetitions exist: the number of repetitions is known *i)* before process start, *ii)* at process start, or *iii)* is identified at some point during runtime.

A compact way of modeling *Ex3-a)* is shown by using a loop task in *Ex3-b)*. In contrast to *Ex3-a)*, it is an option to execute the loop task zero times. This is the case since the process variable controlling the number of executions is checked before the task is executed. As defined by BPMN, repetitions are also created by multi-instance activities represented by two task types: *Ex3-c)* illustrates a parallel task (*T3*) where all instances are being executed simultaneously. A sequential task (*T4*) is part of *Ex3-c)*, where all instances are executed successively. All of these tasks are compact representations of repeating the tasks numerous times without the use of gateways ([125] p. 36f.).

Translating *Ex3* into a resilience graph is straightforward. Since there is no unreliable communication involved in any of the tasks of *Ex3*, the repetitions can be simplified as ordinary tasks. The resulting graph is illustrated in Figure 4.9.

When dealing with unreliable communication, the translation becomes more complex. In general, every possible repetition number of a loop is reflected by a separate

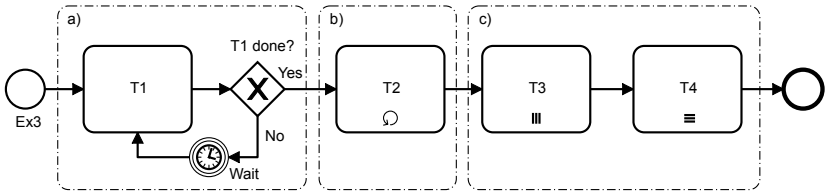

Figure 4.8: Process example *Ex3* featuring loops and multi-instance activities (modeled in BPMN).

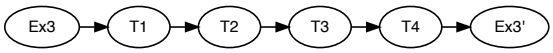

Figure 4.9: Resilience graph of *Ex3*.

path in the graph. Integration of varying resilience statements of the different paths is realized by edge weights. The procedure of calculating edge weights differentiates between connectivity characteristics and connectivity probabilities (cf. section 3.6.1, p. 60ff.). Besides, it depends on whether or not additional communication resources are available for the execution of repetitions. Process example *Ex4* depicted in Figure 4.10 is used to illustrate the translation procedure. At first, graphs based on connectivity characteristics are considered.

In *Ex4*, every task communicates with a participant to realize its functionality. Connectivity between all participants is unreliable. Figure 4.11 depicts the translated graph for *Ex4-a)* and *Ex4-b)* under the condition of up to four executions of *T1* ($\Rightarrow$ maximum repetition number of three). Hence, there are four separate paths for calling *T1*. The number of executions is indicated in parentheses within the corresponding graph vertex (cf. Figure 4.11).

The question of whether or not additional communication resources for the execution of repetitions are available depends on process context. In this example, *T1* of *Ex4-a)* is requesting status information from *P1* periodically. Since time elapses between repetitions, the communication resources are available again for the following repetition. The resilience statements of the message flows remain unchanged and are directly applied to all repetition paths in accordance with Equation 4.2. If zero executions of *T1* would be possible, this path would be assigned with the maximum edge weight $R_{e_{max}}$ since it is always resilient.

$$R_{e_i} = \begin{cases} R_{e_{max}} & \forall\ i = 0 \\ R_e & \forall\ i > 0 \end{cases} \tag{4.2}$$

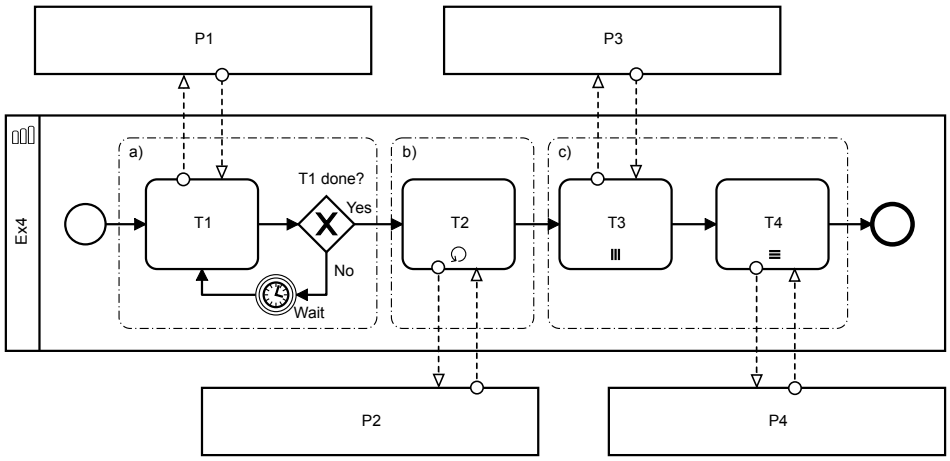

Figure 4.10: Process *Ex4* with loops and multi-instance activities communicating with participants (*rBPMN*).

In contrast to *Ex4-a)*, the loop task of *Ex4-b)* is used to improve an approximation of a value with each of its repetitions. Here, only a single time frame is available to finish the approximation. All repetitions have to share the resources available within this time frame. This results in Equation 4.3 for calculating the repetition-based edge weights. The message flow resilience statements calculated for one task execution are divided by the number of executions. Figure 4.11 illustrates the graph with varying edge weights using Equation 4.2 for *Ex4-a)* and Equation 4.3 for *Ex4-b)*. Since the approximation value may already have its required accuracy when entering *T2*, there is also a path for zero executions of *T2*. No communication with a participant is required in this case. Since this path is always resilient, the maximum edge weight allowed for the scenario (here 4.0 as an example) is used.

$$R_{e_i} = \begin{cases} R_{e_{max}} & \forall\ i = 0 \\ \frac{R_e}{i} & \forall\ i > 0 \end{cases} \tag{4.3}$$

Two multi-instance activities are part of *Ex4-c)* in Figure 4.10. A minimum of one and a maximum of three executions of *T3* and *T4* have been defined for the example. *T3* is a parallel task (three vertical lines in task symbol), hinting that the resources have to be shared for all instances executed in parallel. In contrast, *T4* sequentially executes the instances one after another (three horizontal lines in task symbol). If not stated otherwise by the modeling domain expert, it is expected that *T3* has none and *T4* has additional communication resources for every iteration available. This results in the graph of Figure 4.12 for *Ex4-c)*.

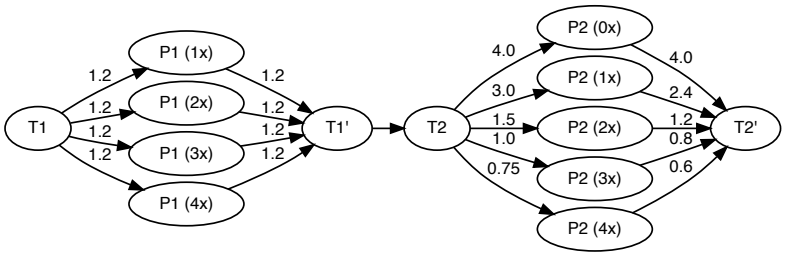

Figure 4.11: Resilience graph of *Ex4-a)* and *Ex4-b)* with connectivity characteristic edge weights.

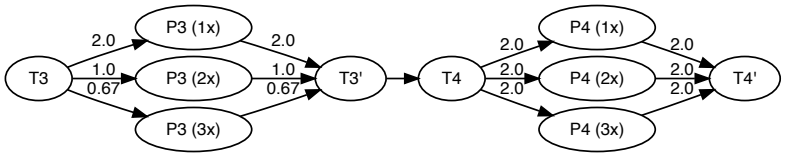

Figure 4.12: Resilience graph of *Ex4-c)* with connectivity characteristic edge weights.

After the graph translation of *Ex4*, some paths are redundant and may be removed. In particular, this applies to the repetition paths for tasks where additional communication resources are available (e.g., *T1* and *T4*). However, the following chapter 5 (p. 95ff.) illustrates that it may be useful for multi-criteria scenarios to integrate and keep redundant paths during graph translation.

Translating repeating process segments which share communication resources can be challenging. If the maximum number of repetitions is unknown, translation of a process model would result in an endless graph. The graph would feature an unlimited number of separated paths for possible repetitions. Translating such a process requires heuristics to simplify graph creation, including a reasonable number of separated paths for repetitions. Heuristics have to be chosen carefully since their quality directly affects the resilience statements identified by graph algorithms.

Knowledge about typical process behavior can be gathered from statistics collected in previous executions of the process model. This allows to significantly reduce the number of graph paths. For instance, it may be adequate to include two paths with the following heuristic: a path for the average number of repetitions and a low-performance path based on the maximum repetition number of the best 90 percent of executions. Statistics should originate from the same scenario that is intended to be executed, due to different behavioral patterns in other scenarios. If no statistics are available, domain experts need to estimate scenario-driven repetition numbers.

The use of connectivity probability weights is illustrated using the graph for *Ex4-c)* in Figure 4.13. While Equation 4.4 defines the calculation of edge weights for repetitions

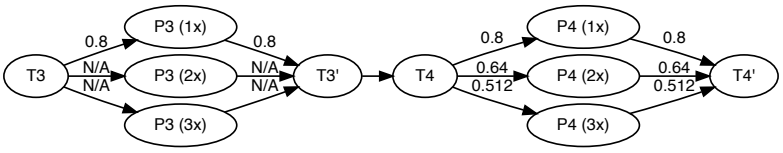

Figure 4.13: Resilience graph of *Ex4-c)* with connectivity probability edge weights.

with additional communication resources available, a calculation for the execution of all instances within the same time frame is not defined (cf. Equation 4.5). Since probabilities are estimated or based on statistics, there is no formula available to estimate connectivity when requesting several times more resources than originally demanded. Domain experts have to define new probabilities for these scenarios.

$$R_{e_i} = \begin{cases} 1 & \forall\ i = 0 \\ {R_e}^i & \forall\ i > 0 \end{cases} \tag{4.4}$$

$$R_{e_i} = \text{N/A (not defined)} \tag{4.5}$$

Whether or not there are additional communication resources for repetitions available is based on the process context and scenario definition. The use of some modeling elements may provide hints (e.g., parallel and sequential multi-instance activities), but is no guarantee for the intention of the domain expert. It is usually not identifiable which case applies for repeating process segments without the help of a domain expert. *rBPMN* introduces the concept of *RepetitionInfo* to still be able to translate a process into a graph. *RepetitionInfo* allows domain experts to state minimum, average and maximum repetition numbers. Besides, there is a flag stating the execution of repetitions in a single time frame. Calculation of edge weights used in repetition path segments may be tricky. Figure 4.14 provides a guideline for calculating edge weights based on connectivity characteristics and connectivity probabilities.

An optimization of resilient operation is possible by dynamically adjusting the graph at process initialization or process runtime. In the first case, autonomous process-to-graph translations and analyses are required. The latter case indicates the need for repeating executions of the resilience analysis and the dynamical adaption of the chosen process path (cf. section 6.2, p. 131ff.).

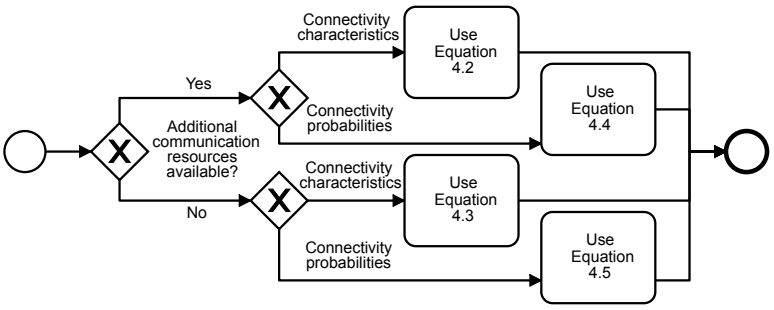

Figure 4.14: A guideline to calculate edge weights for loops and multi-instance activities (modeled in BPMN).

Path Merging by Exclusive Gateways

Parallel and inclusive gateways split the BPMN process token into as many tokens as outgoing gateway options apply. When the tokens arrive at a merging point realized by a parallel or inclusive gateway, the process is synchronized by merging all previously created token copies. However, if merging is realized by an exclusive gateway, no synchronization takes place and all token copies continue their way to the process end. This is also affecting resilience since the merged segment of the process will be executed multiple times.

Similar to the translation of loops and multi-instance activities, it is significant whether or not additional communication resources are available for the merged process segment. In the default case, there are additional resources available. Since the operation of the split path segments differs in their duration, there is a high chance that not all instances of the merged segments are executed at the same time. Even if this is the case resulting in longer execution times, the situation may be reasonable for many scenarios. However, a domain expert may want to share resources along with all merged segment instances for certain scenarios. This can be realized in *rBPMN* by annotating the activities with *RepetitionInfo* metadata.

Process example *Ex5* in Figure 4.15 illustrates process merging by an exclusive gateway. In *Ex5-a)*, a parallel gateway splits the process into two path segments calling services at the participants *P1* and *P2*. All instances of *T3* shall share the available communication resources. Initially, all *OppMessageFlows* have connectivity characteristic edge weight values of 1.2. The resilience graph for *Ex5-a)* shown in Figure 4.16 has been created using parallel gateway translation rules and by calculating edge weights for two executions of *T3* (cf. Table 4.4). After an exclusive gateway merging, *P3* is executed twice since the two process tokens created by the parallel gateway have not been synchronized. Hence, the edge weights connected with *T3* are divided, resulting in values of 0.6.

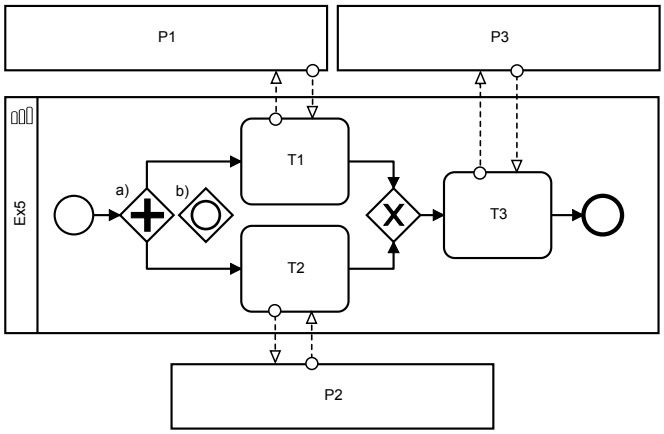

Figure 4.15: Process example *Ex5* featuring a merging exclusive gateway (*rBPMN*).

Figure 4.16: Resilience graph of *Ex5-a)*.

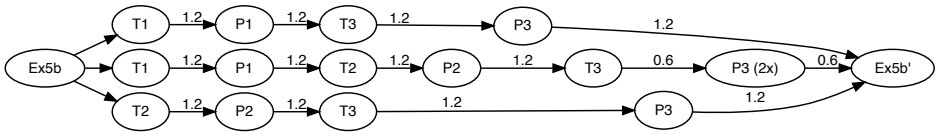

Figure 4.17: Resilience graph of *Ex5-b)*.

The parallel gateway has been replaced by an inclusive gateway in *Ex5-b)*. One, some or all outgoing process segments may be chosen by the inclusive gateway, depending on process variables. The resilience graph in Figure 4.17 reflects this by three separate paths for *i) P1* (top path), *ii) P1* and *P2* (middle path), and *iii) P2* (bottom path). Each path continues to the end of *Ex5-b)* and includes the merged process segment. However, only the resilience for communicating with *P3* is reduced for the case of two token copies (for *P1* and *P2*). If either *P1* or *P2* is executed, only one instance of the path segment calling *P3* is created, having all resources on hand.

Integration of Subgraph Segments

Pools represent participants in BPMN. A participant may be an actor or system part of a different organization, resulting in Service Level Agreements (SLAs) taking care of the resilience of offered services or functionalities. However, a participant may also be part of the same organization accepting resilient configuration demands. In the

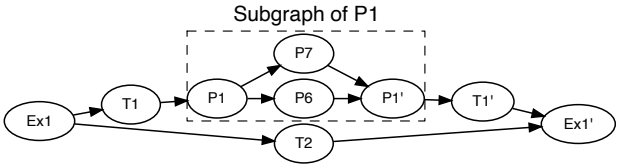

Figure 4.18: Integrated path segment of *P1* into resilience graph of *Ex1*.

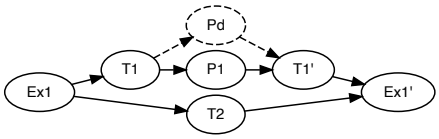

Figure 4.19: Dynamic participant *Pd* in the resilience graph of *Ex1*.

latter case, relevant resilience graph segments of the service-offering participant can be integrated into the graph of the calling participant.

This has been done for the graph of *P1*, which has been inserted into the graph of *Ex1* in Figure 4.18. *P1* includes two separate paths for calling *P6 / P7* and is now part of *Ex1's* resilience calculation. *Ex1* may choose the service for *P1* to enhance resilience. The same procedure allows integrating graphs of sub-processes.

Integration of Dynamic Participants

rBPMN allows to enhance resilience by dynamically integrating participants at process runtime. Using *OppDynTasks*, dynamic participants add an option to an existing *OppMessageGroup* that needs to be reflected in the resilience graph. The illustration provided in Figure 4.19 is based on process example *Ex1* of Figure 4.3. Besides calling a service at *P1*, *T1* may call a service located at the dynamically appearing participant *Pd*. Typical use cases for dynamic participants are ad-hoc, delay-tolerant or opportunistic networks (cf. [66] and [135]).

Preparation of *OppPriorityFlows* for Graph Analysis

The translation of alternatives that are defined as *OppPriorityFlows* equals the translation of *OppDecisionFlows*: every *OppPriorityFlow* being part of an *OppMessageGroup* is reflected by a separate path in the resilience graph. An example is illustrated by process *Ex6* depicted in Figure 4.20 and the corresponding resilience graph in Figure 4.21.

However, a graph preparation regarding path segments including *OppPriorityFlows* is required before the application of a graph analysis. In contrast to *OppDecisionFlows*, a decision is not made based on defined characteristics of participants or activities.

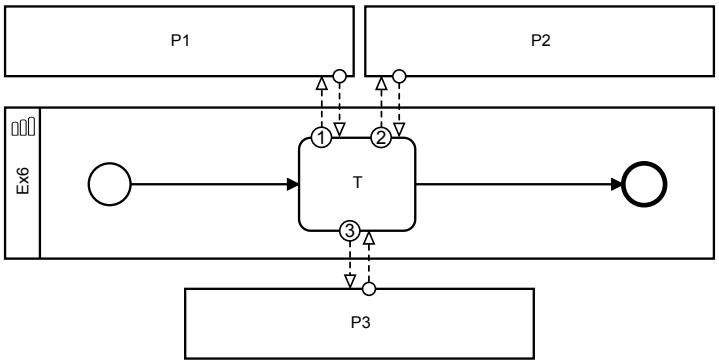

Figure 4.20: Process example *Ex6* with decision-making based on *OppPriorityFlows* (*rBPMN*).

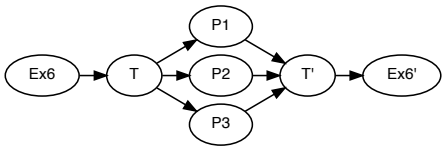

Figure 4.21: Resilience graph of *Ex6*.

Figure 4.22: Resilience graph of *Ex6*, dynamically updated for a graph analysis.

Instead, a domain expert decides on the alternatives by assigning priorities to message flows. Hence, only the highest-ranked alternative satisfying the QoS requirements needs to be part of the resilience graph. Since the availability of alternatives may change, the graph needs to be updated with the highest-ranked resilient alternative dynamically at runtime. This ensures respecting the choice of the domain expert during the graph analysis.

In regards to the resilience graph of *Ex6*, only one alternative remains in the graph before a resilience analysis is applied. For instance, if *P1* is not available, only the path of *P2* would remain in the resilience graph of *Ex6*, as depicted in Figure 4.22. This way, a graph analysis follows the domain expert's choice and is unable to choose the path including *P3*, although it might be more resilient compared to the path of *P2*.

4.3 Resilience Graph Analysis

The resilience graph is the foundation of the resilience analysis. The graph edge weights represent the resilience values of message flows, calculated by the approach introduced by *rBPMN* (cf. section 3.6, p. 59ff.). The literature describes numerous algorithms applicable to DAGs, aiming to solve different problem statements (cf. section 2.3, p. 31f.). The well-known SPF problem of finding a graph path with minimized cost shows some similarities to the resilience problem and is considered subsequently. The SPF calculations are basically a speedup technique for an all-paths graph analysis, which is also considered for finding and ranking resilient process paths. Besides, a heuristic for highly dynamic scenarios and a novel approach to combine process paths that use different communication technologies are introduced subsequently.

4.3.1 SPF and LPF Analysis

SPF algorithms such as Dijkstra [65] and Bellman-Ford [10] find the path with the lowest total weight (also known as cost) from a source to a destination. Since not minimum cost, but maximum resilience is desired here, edge weights need to be adjusted either by *i)* inverting positive weights ($R_{e_{new}} = (R_{e_{old}} - R_{e_{max}}) * (-1)$), by *ii)* using the reciprocal value ($R_{e_{new}} = R_{e_{old}}^{-1}$) or by *iii)* applying negative weights ($R_{e_{new}} = R_{e_{old}} * -1$), if supported by the SPF algorithm. The three options are equivalent and do not affect the result of an analysis in terms of resilient operation.

Alternatively, a longest-path analysis identifying the path with maximum total weight is applied. While the so-called *longest-path-first (LPF)* problem is NP-hard in general, it can be solved in polynomial time in DAGs applied here (cf. [154]). If calculation effort is not critical, implementations may be based on the Breadth-First Search (BFS) [65]. The LPF method is used for subsequent explanation examples.

Figure 4.23 shows the result of an LPF analysis on process example *Ex2* with the chosen path as a dashed line. Exemplary connectivity characteristic edge weights have been applied. While the path with the highest resilience R_t from *Ex2* $\rightarrow$ *Ex2'* has been chosen, the path is not resilient because of the path resilience level $R_l = 0.8$. This is due to the impact of an edge with a weight of 6.0, which guides the algorithm to include *P1* into the path. The mechanism might help for maximizing the total weight, but not for finding resilient process paths.

Removing all non-resilient edges $R_e < 1$ (cf. Equation 4.1) from a graph enables the analysis to find only resilient paths. Furthermore, it is suggested to limit the maximum weight of an edge $R_{e_{max}}$ to avoid over-weighting graph edges. For this thesis, *over-weighting graph edges* is defined as a condition where edges with high weight values have a dominating impact on the graph. Due to their high values, the graph analysis

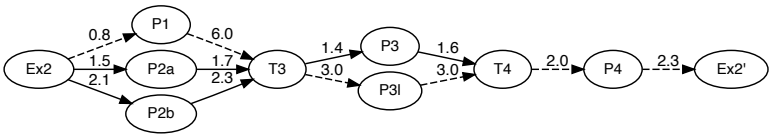

Figure 4.23: LPF analysis on *Ex2* using estimated connectivity weights.

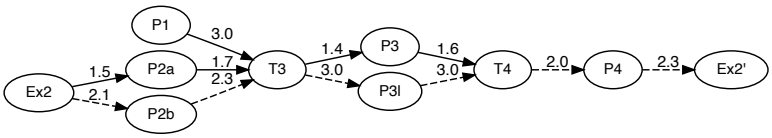

Figure 4.24: LPF analysis on *Ex2* after removal of non-resilient edges ($R_{e_{max}} = 3.0$).

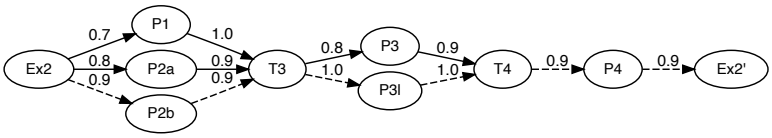

Figure 4.25: LPF analysis on *Ex2* using probability edge weights.

becomes biased since other edges become irrelevant. $R_{e_{max}}$ is also applied to edges representing locally moved functionality (e.g., edges connected to *P3l* of *Ex2*) and edges that have no influence on resilience. Figure 4.24 presents an adjusted resilience graph and the resulting path with the highest total resilience R_t of an LPF analysis.

Figure 4.25 illustrates an LPF analysis using exemplary probability values as edge weights. The analysis summarizes edge weights to choose the path with the highest total weight $\sum_{i=1}^{n} P_{e_i}$. Especially when using probabilities, this might not identify the most appropriate path. It is important to include other metrics such as the probability of the path P_p. Furthermore, adjusting the graph by removing all edges not fulfilling a defined minimum resilience level P_l may be useful.

4.3.2 SPF Speedup Techniques

Graph adjustments have to be made when using SPF or LPF algorithms on a resilience graph. The effort of these adjustments is usually negligible in terms of required performance. For instance, the removal of non-resilient/unqualified graph edges programmatically kept in a list has minimal performance impacts. However, adjustments can be avoided by using *constrained SPF (CSPF)* algorithms. These algorithms only consider graph edges that respect defined constraints. The constraints need to ensure the avoidance of unqualified edges in selected paths [183].

Since the original SPF algorithm, many variations have been developed for different purposes (cf. section 2.3.1, p. 31f.). A major area of research is the field of SPF speedup variants. For instance, the *A* algorithm* tries to improve calculation speed by using heuristics [85]. *Contraction hierarchies (CH)* allow accelerated calculations by adding shortcuts to the graph in a preprocessing phase [183]. Algorithms of the *k-shortest-paths (KSP)* type allow identification of the first k resilient paths [63], serving as alternatives/backups in the case of dynamic environments. Many more SPF variants exist in the literature.

In general, the speedup techniques are applicable to the resilience problem. However, their positive effect on path calculation times is questionable. Most speedup techniques focus on very large graphs as CH algorithms do. They originate from road networks, where it is common to have thousands or millions of vertices and edges in a graph. This is very unlikely for resilience graphs, even if a graph includes numerous subgraphs of other participants.

4.3.3 Maximum-Step Analysis

With the *maximum-step (Max-Step)* analysis, a straightforward heuristic for finding resilient paths in processes with a single endpoint is introduced. The algorithm chooses the edge with the highest weight at each decision point (or step) until the end vertex is reached. The algorithm includes an optional parameter for a minimum path resilience level R_l. If it has to choose an edge not meeting the desired resilience level, the algorithm returns to the last decision point and chooses the next highest-ranked edge to continue the path to the end vertex. The analysis will present no outcome if no path with the required resilience level exists.

Applying the maximum-step heuristic to the *Ex2* graphs of Figures 4.24 and 4.25 leads to the same chosen paths as depicted in the Figures. Facing conditions as shown by Figure 4.26 is challenging for the maximum-step analysis and will not result in finding the most resilient path. In the graph of Figure 4.26, the heuristic is unable to notice the more resilient path including *P2 / P4 / P6*. However, the algorithm may be an appropriate choice for highly dynamic scenarios featuring low amounts of vertices, where graph edge values show a level of uncertainty or do change rapidly. The algorithm is also a solid choice for scenarios including a low number of separated paths.

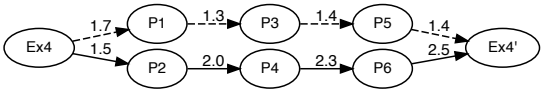

Figure 4.26: Challenging resilience graph for the maximum-step heuristic.

4.3.4 All-Paths Analysis

Evaluating resilience graphs using the *all-paths* method is a versatile type of analysis. All possible process paths between source and destination are identified (e.g., using BFS or Depth-First Search (DFS)) [64]. The different paths are compared by use-case-driven metrics. For instance, the resilience level of a path R_l is of main interest for many scenarios, especially when distinguishing between different resilient process paths. When working with non-resilient process paths and edge weights based on probabilities, the probability of the path P_p might be the most relevant factor.

Table 4.6 compares the paths for the process example *Ex2* with different metrics. The connectivity characteristic (CC) and connectivity probability (CP) weights of Figures 4.23 and 4.25 are used for the calculations. Evaluating the graph by using the total resilience R_t leads to choose the path including *P1 / P3l*, a non-resilient process path. Focusing on the resilience level of the path R_l corrects the evaluation by choosing the path including *P2b / P3l*. The same result is presented by the probability of the path P_p. Metrics may be prioritized or combined to decide on the most appropriate process path.

4.3.5 Combined-Paths Analysis

Finding the most resilient configuration of a process does not necessarily mean to find the single path with the highest resilience level R_l or the highest probability P_p, depending on whether connectivity characteristics or connectivity probabilities are used. A process may utilize hybrid networks to combine different technologies for communication [111]. For instance, the combination of infrastructure-based (e.g., Cellular, WiFi in access-point mode, LoRaWAN) and infrastructure-free (e.g., WiFi in ad-hoc mode)

Table 4.6: Comparison of different process paths of *Ex2* using an all-paths analysis and selected metrics.

Path variation	**CC edge weights**				**CP edge weights**			
Ex2 → Ex2'	R_l	R_t	R_a	R_r	P_p	P_l	P_a	P_r
P1 / P3	0.8	14.1	2.35	5.2	0.41	0.7	0.87	0.3
P1 / P3l	0.8	17.1	2.85	5.2	0.57	0.7	0.92	0.3
P2a / P3	1.4	10.5	1.75	0.9	0.42	0.8	0.87	0.1
P2a / P3l	1.5	13.5	2.25	1.5	0.58	0.8	0.92	0.2
P2b / P3	1.4	11.7	1.95	0.9	0.47	0.8	0.88	0.1
P2b / P3l	2.0	14.7	2.45	1.0	0.66	0.9	0.93	0.1

technologies multiplies communication opportunities, especially in environments with unreliable communication (cf. section 2.1, p. 10ff.).

The *combined-paths (Com-Paths)* analysis introduced in this subsection allows combining process paths realized by different communication technologies to enhance the resilience of a process. The analysis is inspired by maximum-flow algorithms of network graphs such as [70] and [46], finding the maximum amount of flow able to be transferred from a source to a sink in a capacity-restricted network. While resilience graphs already represent network graphs, maximum-flow algorithms are not applicable here: Resilience graphs do not use capacity-labeled edges required by the algorithms, but edge weights based on connectivity characteristics or connectivity probabilities. The meaning of edge weights in a resilience analysis and in a maximum-flow analysis is not equivalent and the application of maximum-flow algorithms will not necessarily result in enhanced resilience. For instance, a combination of non-resilient connectivity characteristic weights of different paths does not result in a resilient path. However, the maximum-flow principle may be adapted for resilience as shown subsequently. A prerequisite for using a combined-paths analysis on a resilience graph is to only include separate path segments that use different communication technologies compared to each other.

Figure 4.27 illustrates the use of different technologies to communicate with participants in *Ex2* with connectivity probability edge weights. A combination of paths is possible for segments *Ex2* $\rightarrow$ *T3* and *T3* $\rightarrow$ *T4*. Combining m different paths to a joint edge P_{je} requires to identify the overall probability of every path first. Equation 4.6 calculates the probability of a path P_p by multiplying its consecutive edge weights P_e (cf. Table 4.3). Following, the probabilities of the m different parallel path segments are combined to a joint edge weight using Equation 4.7.

$$P_p = \prod_{i=1}^{n} P_{e_i} \tag{4.6}$$

$$P_{je} = 1 - \prod_{j=1}^{m} (1 - P_{p_j}) \tag{4.7}$$

Consequently, a graph with combined segments as shown in Figure 4.28 is created. Two joint edges for *Ex2* $\rightarrow$ *T3* (0.98) and *T3* $\rightarrow$ *T4* (1.0) are included in the combined paths graph. Based on this graph, metrics such as the path probability P_p and the path resilience level P_l may be applied to evaluate effectiveness of path combinations. For instance, evaluation of the path probability P_p of the most resilient path of Figure 4.27 (including $P2b$ and $P3l$) and the combined path of Figure 4.28 illustrates an enhancement from $P_{p_{F4.27}} = 0.66$ to $P_{p_{F4.28}} = 0.79$.

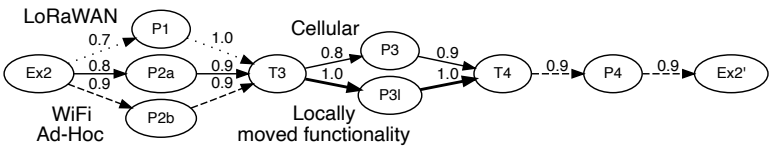

Figure 4.27: Enhancing resilience by combining communication technologies.

Figure 4.28: Combined resilience of different communication technologies.

Applying the combined-paths analysis on graphs with connectivity characteristic weights is not advised. A calculation to combine path segments similar to Equation 4.8 potentially leads to a false statement about process resilience. The calculation summarizes the average weight of every path segment to a joint edge weight R_{je} and may include weak or unstable edges (cf. *Ex2* $\rightarrow$ *P1* $\rightarrow$ *T3* in Figure 4.23). However, Equation 4.8 may help to enhance the resilience of processes by demonstrating the outcome of a parallel process path execution. Prior removal of non-resilient graph edges will prevent false resilience statements.

$$R_{je} = \sum_{j=1}^{m} \left(\frac{\sum_{i=1}^{n_j} R_{e_{j,i}}}{n_j} \right) \tag{4.8}$$

4.3.6 Comparison of Graph Algorithms

With a resilience graph on hand, a variety of graph algorithms becomes available for analyzing resilient operation. Table 4.7 summarizes the characteristics of algorithms considered in this thesis. The table states whether or not a graph needs preparation before applying an algorithm, e.g., removing non-resilient edges. It is outlined whether or not the algorithm is a speedup technique, if it considers the complete path for its decision and if it is capable of finding multiple path.

Depending on the applied algorithm, it is not only possible to find the most resilient path but to compare and rank the resilience of different process paths. Some algorithms allow optimizing the resilience of path segments by combining paths that use distinct communication technologies. Others identify alternative paths / backup paths in case of a dynamically changing environment.

The ability to directly include decision points of subsequent process parts is of main importance. Algorithms that are complete (cf. Table 4.7) consider the whole path for a decision, not only the current decision point in the process. The decision-making for

Table 4.7: Categories of graph algorithms and their applicability for the graph weight types connectivity characteristics and connectivity probabilities.

Graph weight type	*SPF*	*CSPF*	*LPF*	*KSP*	*A**	*CH*	*Max-Step*	*All-Paths*	*Com-Paths*
Connectivity characteristics	✓	✓	✓	✓	✓	✓	✓	✓	-
Connectivity probabilities	-	-	-	-	-	-	✓	✓	✓
Algorithm characteristics									
Graph preparation required?	✓	-	✓	✓	✓	✓	-[1]	-	✓
Speedup technique?	✓	✓	✓[2]	✓	✓	✓	✓	-	-
Alternative path(s) identification?	-	-	-	✓	-	-	-	✓	-
Completeness?	✓	✓	✓	✓	✓	✓	-	✓	✓

Declaration:
[1] preparation required if only resilient paths shall be found
[2] depending on the applied algorithm

a path can be automated using graphs and the scenario-driven selection of resilience metrics.

Resilience may benefit from the simultaneous use of graph algorithms. For instance, an all-paths analysis may help domain experts to get familiar with the characteristics of a scenario at design time. Typical process behavior and useful resilience metrics can be identified. Weak and non-resilient parts of the process model can be optimized. Afterward, SPF algorithms may speedup the decisions at runtime. This might be especially useful for time-critical or performance-restricted scenarios. Also, a speedup technique may be a reasonable choice for highly dynamic scenarios.

A graph analysis can identify dynamic changes at runtime. Often, an early identification of resilience issues in upcoming process segments allows avoiding process failures by adapting the path accordingly. Information about the current connectivity within a scenario may be shared along participants, allowing every participant to extend their knowledge and to update their graphs for decision-making. The knowledge gained by a graph analysis at design time may instruct alternative decision-making techniques with its configuration for process runtime, if graphs should not be used at runtime.

The graph-based approach of a resilience analysis can be extended to include other process criteria. This is illustrated in the following chapter 5 (p. 95ff.).

Summary

This chapter introduces a graph-based approach for the resilience analysis of business processes. A set of resilience metrics allows to observe different characteristics of a process path and to compare paths against each other.

Before the resilience of a process can be verified, its model needs to be translated into a DAG in the form of a resilience graph. A rule set guides the translation procedure from process-to-graph, including examples for inclusive gateways, loops, local functionality, merging exclusive gateways and dynamic participants.

Various types of graph algorithms are available for the identification of resilient process paths. SPF algorithms are commonly used to find the path with minimum cost. While SPF can be mapped to find resilient paths, the graph has to be prepared avoiding the integration of non-resilient edges. Several SPF speedup techniques are available for time-critical or performance-restricted scenarios. The all-paths analysis allows the inspection of all possible process paths with selected metrics. Process resilience may be optimized by using the combined-paths analysis, using different communication technologies in parallel.

CHAPTER 5

GRAPH-BASED MULTI-CRITERIA ANALYSIS

The resilience of communication is a major topic for processes taking place in unreliable communication environments. However, resilience is not the only criterion of importance for many scenarios. Process criteria such as the operation accuracy of activities, the required cost for calling services at participants, and the time needed for a process path are often relevant factors. The question arises of how these criteria can be combined to find a common process path, fulfilling the demands of the criteria set.

Therefore, this chapter outlines the graph-based multi-criteria analysis of business processes following the process steps depicted in Figure 5.1. Before presenting the process steps in detail, the chapter introduces criteria-based metrics to evaluate process paths. Following, a classification and prioritization scheme for the set of chosen criteria and adapted process-to-graph translation rules are elaborated. Criteria can be assigned to a translated graph separately, jointly, or a mixture of both. Since a variety of graph algorithms applicable for the multi-criteria analysis are outlined in the literature, the focus of the graph analysis section is to elaborate different approaches for the use of existing algorithms. The objective of the approaches is to provide a wide range of analysis options, satisfying the needs of various scenarios and application domains. This thesis introduces approaches based on iterations, on comparison, on multi-criteria graph algorithms as well as scenario-based combinations of the first three approaches.

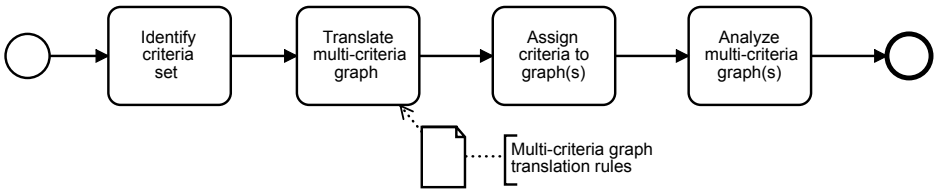

Figure 5.1: Process steps of a multi-criteria analysis (modeled in BPMN).

5.1 Criteria Metrics

This thesis defines criteria metrics in the form of C_y^x, where y represents a certain metric (e.g., t for total path weight) and x stands for a concrete criterion (e.g., R for resilience). For instance, C_t^R describes a total path weight for resilience while C_w^A represents an accuracy value ($\Rightarrow A$) used as an edge weight ($\Rightarrow w$). If probabilities are used as edge weights, the format changes to P_y^x with probability edge weights $P_w^x \in \mathbb{R} | 0 \leq P_w^x \leq 1$. The metrics introduced in this thesis for the analysis of criteria graphs are listed in Table 5.1.

While the usefulness of the different metrics depends on the applied use case, the total path weight C_t^x and the lowest or highest path weight C_l^x and C_h^x are of importance

Table 5.1: Criteria metrics.

Semantic	**Symbol & Formula**	**C**	**P**
Criterion placeholder	x (e.g., R $\Rightarrow$ Resilience)	✓	✓
Number of weighted path edges	$n \in \mathbb{N}$	✓	✓
Edge weight	$C_w^x \in \mathbb{R}$	✓	
Total path weight	$C_t^x = \sum_{i=1}^{n} C_{w_i}^x$	✓	
Lowest path weight	$C_l^x = min(C_{w_1}^x, ..., C_{w_n}^x)$	✓	✓
Highest path weight	$C_h^x = max(C_{w_1}^x, ..., C_{w_n}^x)$	✓	✓
Average path weight	$C_a^x = C_t^x / n$	✓	✓
Median path weight	$C_m^x = median(C_{w_1}, ..., C_{w_n})$	✓	✓
Range of path weights	$C_r^x = C_h^x - C_l^x$	✓	✓
Probability of edge weight	$P_w^x \in \mathbb{R} \| 0 \leq P_w^x \leq 1$		✓
Probability of path	$P_p^x = \prod_{i=1}^{n} P_{w_i}$		✓
Boolean probability of path	$P_b^x = 0 \quad (\forall\, P_p^x < 1)$ $P_b^x = 1 \quad (\forall\, P_p^x = 1)$		✓

Declaration:
C: ✓$\Rightarrow$ applicable for connectivity characteristics
P: ✓$\Rightarrow$ applicable for connectivity probabilities

for most scenarios. With probabilities as edge weights P_w^x, the probability of the path P_p^x is a substantial metric. If a criterion requires a probability of 1.0, the boolean path probability P_b^x is of relevance. For instance, P_b^R states if the chosen path is resilient ($\Rightarrow 1.0$) or non-resilient ($\Rightarrow 0.0$)

The metrics listed in Table 5.1 are related to a single criterion. The following subsections present techniques developed for this thesis to combine criteria metrics for the process analysis and to analyze processes based on distinct metrics. In the latter case, the term *Pareto-optimal* is of interest. A process path is Pareto-optimal if there is no other path available optimizing one criterion without deteriorating another criterion (cf. section 2.3.2, p. 33). A process may include a set of Pareto-optimal paths. The following subsections provide examples for the use of the multi-criteria metrics. Further on, the suitability of Pareto-optimal paths in terms of a multi-criteria analysis including communication resilience is investigated.

5.2 Process Criteria Identification, Categorization and Prioritization

Depending on the concrete process, the intended scenario and the application domain, a broad variety of criteria may be relevant for a process. Naturally, the number of criteria, their meaning for the process and their level of importance vary. While some criteria address optimization of process operation, others may measure side effects, considered as monitoring parameters. Even a set of only optimization criteria needs prioritization in most cases. Some optimization criteria may require to strictly meet the demanded metrics while other criteria aim to optimize the chosen metrics in the best possible way. This illustrates that it is not only relevant to choose/identify a set of criteria for a process but to distinguish criteria against each other by categorizing and/or prioritizing them. Hence, analyzing optimal operation of a process requires to identify, categorize and prioritize criteria against each other and needs to be addressed by the multi-criteria analysis.

An example illustrates the categorization and prioritization of process criteria. The following criteria are chosen:

- **Resilience** (Resil.): Stability of a process against communication issues.
- **Accuracy** (Accur.): Precision of the calculation/operation of an activity.
- **Cost**: Monetary value required to execute an activity.
- **Time**: Time frame required by an activity to finish its operation.

- **Privacy**: Describing the privacy of information processed by an activity.
- **Automation**: Indicating the level of automation for an activity.

It is beneficial for most scenarios to categorize and prioritize the criteria set according to the demands of domain experts. As part of this theses, a three-level classification concept is introduced guiding the classification and prioritization of a criteria set.

For example, domain experts state that the criteria resilience and accuracy are most critical for process operation. In the graph analysis, minimum path edge weights C_l^R and C_l^A will be defined that have to be met as demanded by the experts. Resilience and accuracy represent 1. level criteria, describing the type of 1st tier optimization criteria. Cost and time also represent optimization criteria. For cost and time, the experts' demands are to reduce the total path weights C_t^C and C_t^T as best as possible, without defining maximum path weights. Hence, cost and time represent 2nd tier optimization criteria. Within this 2nd tier, cost is prioritized over time. Finally, experts define privacy and automation as monitoring criteria, not affecting the process path choice.

Table 5.2 summarizes the classification concept using the example criteria set. The concept may be adapted according to the needs of application domains and concrete process scenarios. Use of the concept is illustrated in sections 5.4, 5.5, and 7.1.2 (p. 145ff.).

Most of the time the identification, categorization and prioritization of criteria is performed before the process is translated into a graph. However, it is also possible to move the step of choosing and rating process criteria after the process-to-graph translation. In this case, simplification during the graph translation should be avoided. It may remove path segments relevant for a criterion which may be selected afterward.

5.3 Multi-Criteria Process-to-Graph Translation

This section elaborates the creation of process graphs, building the foundation of the graph-based multi-criteria analysis. A *process graph* is defined as the result of a process-

Table 5.2: Importance levels for the categorization and prioritization of criteria.

Importance level	**1. Level**	**2. Level**	**3. Level**
Level type	Optimization (1st tier)	Optimization (2nd tier)	Monitoring
Semantic: Meeting criteria demands is ...	required	desired	negligible
Example criteria	Resilience Accuracy	Cost (Priority 1) Time (Priority 2)	Privacy Automation

to-graph translation. After applying criteria-based graph edge weights to a process graph in the following section, it is defined as a *multi-criteria graph*.

The majority of translation rules created for the analysis of process resilience (cf. section 4.2, p. 69ff.) are directly applicable for the translation of process models to process graphs. However, process segments including optional message flows and repetitions require special considerations. If additional time is available, repetitions may not affect resilience since no sharing of communication resources is required (cf. section 4.2.3, p. 76ff.). In contrast, other criteria such as the time of process operations are affected, which prevents the elimination of repetitions in the process graph. Also, guidelines for the simplification of graphs need adjustments due to the same reason: Process segments irrelevant for resilience may affect other criteria such as accuracy, cost, or time.

The translation of process models to process graphs is illustrated by using process example *Ex7* depicted in Figure 5.2. The upper process path *Ex7-a)* communicates with *P1* to realize its functionality. The lower path *Ex7-b)* does not require communication with other participants. Applying the translation and simplification rules of section 4.2 (p. 69ff.) to the process model of *Ex7* leads to the resilience graph illustrated in Figure 5.3. *T2* with its repetitions has been omitted and the tasks of *Ex7-b)* have been merged to a single vertex *T*. This is possible due to the missing of communication with other participants. The compact graph of Figure 5.3 is well-suited to analyze communication resilience but is insufficient for the analysis of other criteria. For instance, if the operation accuracy of tasks is a relevant criterion, the repetitions of *T2* and the different paths of *Ex7-b)* are missing in the graph. Repetitions of *Ex7-a)* and path variations of *Ex7-b)* both influence the accuracy of the process.

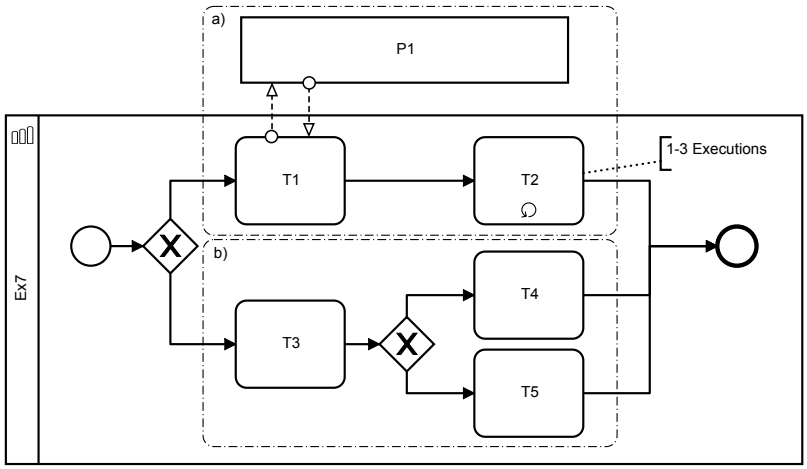

Figure 5.2: Process example *Ex7* (*rBPMN*).

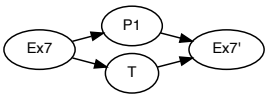

Figure 5.3: Simplified resilience graph of *Ex7*.

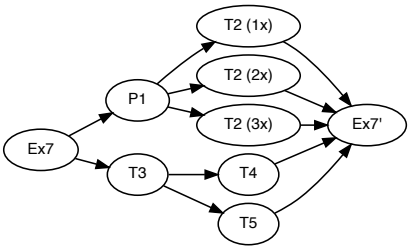

Figure 5.4: Process graph of *Ex7*.

Concerning repeating process segments such as the loop task *T2* in *Ex7*, it is advised to always include all possible path variations at first. This means avoiding any simplifications during the initial multi-criteria process-to-graph translation by adding a separate path for every possible number of executions. This leads to the upper section of the graph presented in Figure 5.4 with three different paths for the possible executions of *T2*. The number stated within the parenthesis of a vertex defines the number of executions of the corresponding activity (e.g., *1x* $\Rightarrow$ one execution). Afterward, repetition-based edge weights may be used for criteria affected by repetitions while non-affected criteria keep the same edge weights for all three paths. Eventually, a graph simplification can reduce the number of paths if non-relevant paths for all chosen process criteria exist.

The same approach is applied for the translation of process segments missing communication with other participants. Instead of an early simplification, it may be useful to initially include all possible paths in the graph and simplify after considering the chosen criteria and their effects. For instance, the translation of *Ex7-b)* results in the lower part of Figure 5.4, which is accurate for criteria such as accuracy and time. When dealing with criteria only applicable to external participants (e.g., resilience and cost), the simplification of summarizing all tasks in a T vertex (cf. Figure 5.3) may be sufficient.

The process-to-graph translation of optional *OppMessageFlows* is a rule that shows major differences regarding the translation of resilience graphs and process graphs. While optional *OppMessageFlows* are irrelevant for the resilience of a process, other criteria are affected by the execution of the called activities. Since these process segments are only optional in terms of process resilience, they need to be considered as

mandatory in terms of other criteria. For instance, the cost of a process execution may raise by the successful execution of an optional message flow calling another participant. It seems reasonable to include the cost for this situation by default since the cost for the uncertain execution needs to be reserved. Further, the accuracy of operation as a criterion may be directly influenced by the number of executed repetitions. Since there is no guarantee for the execution of the optional *OppMessageFlow* and the activities of the called participant, it is advised to exclude possible improvements of the accuracy as a precaution.

This raises the question of how to handle optional *OppMessageFlows* in process graphs. A possibility is to include an additional vertex as part of a separated path for the execution of the optional *OppMessageFlow*. This is illustrated by process example *Ex8* depicted in Figure 5.5 and the corresponding graph in Figure 5.6. This way, the influence of the optional execution is directly identifiable by the chosen criteria metrics. Non-affected criteria may be handled in a neutral way without affecting their metrics. However, the resilience criterion points out that this is not always manageable: In the case of non-inverted connectivity characteristic weights, a weight would have to be added for the optional *OppMessageFlow* (e.g., the maximum edge weight), increasing the total resilience of the separated path. Besides, the approach may imply a choice of one or the other path which does not exist.

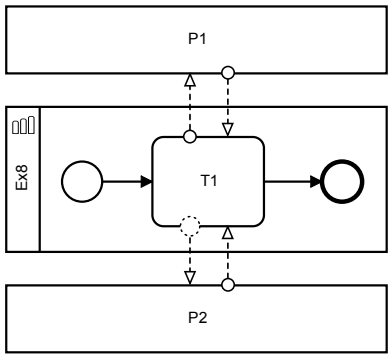

Figure 5.5: Process example *Ex8* including an optional *OppMessageFlow* (*rBPMN*).

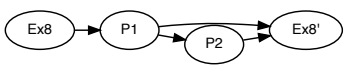

Figure 5.6: Process graph of *Ex8* with separated paths.

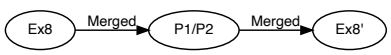

Figure 5.7: Process graph of *Ex8* with merged weight segments.

The graph depicted in Figure 5.7 illustrates an alternate approach. Instead of adding a path for the optional *OppMessageFlow*, edge weights of *P1* are merged with *P2* for affected criteria. For instance, resilience may keep the weights of *P1* and ignore *P2*. In contrast, the cost criterion may summarize costs for *P1* and *P2* while the accuracy criterion may only include weights for *P1*. The approach provides an enhanced level of flexibility for every criterion. Unintended influence on criteria metrics is avoided, no non-existing choices are implied by different paths in the graph. Hence, the approach of merging weights is proposed against adding separated paths for most scenarios.

The situation may be different for the translation of optional *OppPriorityFlows* and *OppDecisionFlows* into graphs. Here, an *OppMessageFlow* may be chosen from a set of alternatives, resulting in different options for the domain expert. The decision is either driven by priorities or by defined characteristics/criteria. With a set of optional alternatives and only an incoming and outgoing non-optional edge available for merging, not all options may be merged into the process graph. Hence, merging weights for the alternatives is a compromise, e.g., by using minimum, average, or maximum weights of the alternatives set. This may be satisfying for criteria that have a rather informational character than an optimization impact on process execution. For the latter case, it is more meaningful to include separate paths for every *OppMessageFlow* provided by the set of alternatives. Since a decision for merging weights or adding separated paths is heavily bound to the specific scenario, both options are possible for the translation of process models to process graphs.

Table 5.3 summarizes the translation aspects discussed in this section in a generally applicable manner. Besides, the rules for the translation of message flows and path merging by exclusive gateways have been updated (cf. Tables 4.4 and 4.5, p. 73f.). The rules remain unchanged but the description was generalized to be applicable for the multi-criteria translation.

Table 5.3: Summary of adapted resilience graph creation rules for the multi-criteria translation of BPMN and *rBPMN* process elements to graph segments.

Process element	**Graph segment**		**Explanation**
BPMN flows / events			
Message flow	Extended path		~~Resilience depends of the success of the *message flow*~~ Criteria depend on the *message flow* or the activities being called.
***rBPMN* flows**			
OppMessageFlow, OppPriorityFlow, OppDecisionFlow	a) Extended path		a) If *opportunistic message flow* is declared as *required*. Caution: respect *OppMessageGroup* rules.
	b) ~~Remove element~~	*	b) If *opportunistic message flow* is declared as *optional*.
	Option 1: Merged weights		Option 1: Weights for the opportunistic segment may be merged with non-opportunistic edge weights, where required by criteria.
	Option 2: Separated paths		Option 2: A separate path for every optional message flow. Non-affected criteria may apply neutral edge weights.
BPMN loops / multi-instance activities			
loop parallel sequential	a) Separated paths ~~b) Extended path~~	*	a) Separate paths for every number of executions. Calculation of repetition-based edge weights required if ~~communication~~ criteria are affected by repetitions. ~~b) If communication resources are not shared / if resilience is not affected.~~
BPMN path merging by exclusive GW			
Splitting GW: parallel, inclusive, parallel event-based, complex	Separated and extended paths	*	Splitting GW and merging exclusive GW result in multiple executions of the merged process segment. Calculation of repetition-based edge weights required if ~~communication~~ criteria are affected by the multiple executions.

Declaration: * provisioning of process context information eliminates semantic gap

5.4 Multi-Criteria Graphs

The previous section presents an updated rule set for the translation of process models to process graphs. As a next step, this section illustrates how to apply different criteria to the created DAG. With separate graphs, joint graphs, and multi-dimensional graphs, three approaches are presented for the application of criteria to process graphs. The focus of this section is to point out the differences of the approaches, and to illustrate the aspects needed to be considered regarding the application of graph edge weights. The concepts of separate and joint graphs have been developed as part of the multi-criteria analysis approaches of this thesis. The techniques to merge edge weights for joint graphs are based on literature, as well as the concept of multi-dimensional graphs.

The three approaches of *separate*, *joint*, and *multi-dimensional graphs* are illustrated using process example *Ex9*. *Ex9* has been divided into the process segments *a)*, *b)* and *c)*, depicted in Figure 5.8. Starting with *Ex9-a)*, a decision has to be made for a path using either *T1* or *T2*. While the path of *T1* includes multiple following tasks and communication transactions with participants, *T2* directly leads to the process endpoint without any further activities or communication. The additional choices part of *Ex9-b)* and *Ex9-c)* only appear if *T1* has been chosen. The corresponding DAG has been created using multi-criteria translation rules and is presented in Figure 5.9. The graph of *Ex9* includes two glue vertices *G1* and *G2*, combining the segments between *Ex9-a)* and *Ex9-b)* as well as *Ex9-b)* and *Ex9-c)*. *T4* is executed once, twice, or three times. These repetitions are also reflected in the graph of Figure 5.9.

With the exemplary criteria set of section 5.2, six different criteria are chosen. Resilience and accuracy for 1st tier process optimization, cost and time for 2nd tier optimization and privacy and automation as monitoring criteria (cf. Table 5.2). After the identification, categorization and prioritization of the process criteria set, the criteria are applied to the process graph by adding criteria-based graph weights. A weight is used to express how well a criterion is met by the related process element. Depending on the concrete criterion and process element, a weight may belong to an incoming edge of a vertex, an outgoing edge of a vertex or to the actual vertex itself. Taking the call of a service at *P1* in *Ex9* as an example, criteria for accuracy, cost, time, privacy and automation are all related to the actual vertex of *P1*. This is due to the criteria describing characteristics of the service offered by *P1*. However, the resilience criterion describes stability of communication with the service. Resilience weights belong to the incoming and outgoing edges of *P1*, reflecting *OppMessageFlows* for a request to and a reply from the service.

Before the application of weights to edges and vertices, a verification is required to check if the intended graph analysis algorithms support all applied weight types. Since

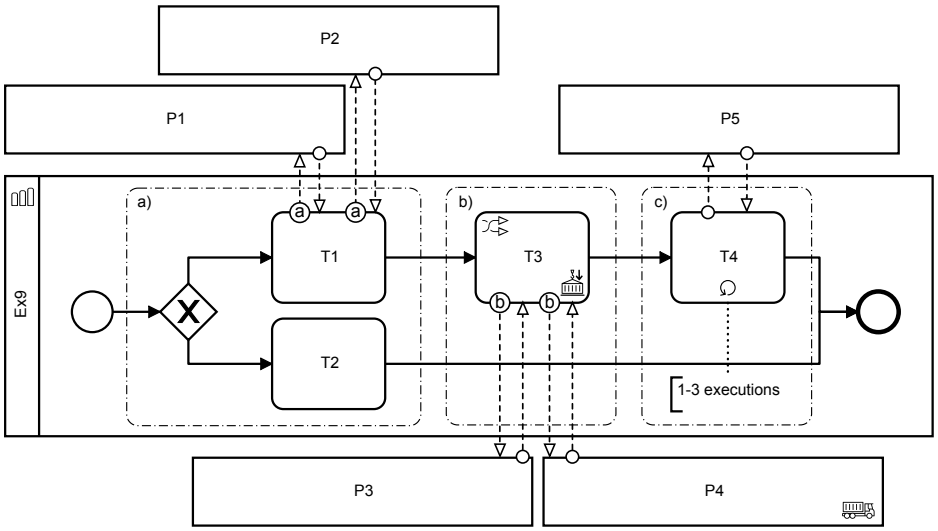

Figure 5.8: Process example *Ex9* including segments *a)*, *b)* and *c)*, serving as a basis for the application and analysis of multi-criteria graphs (*rBPMN*).

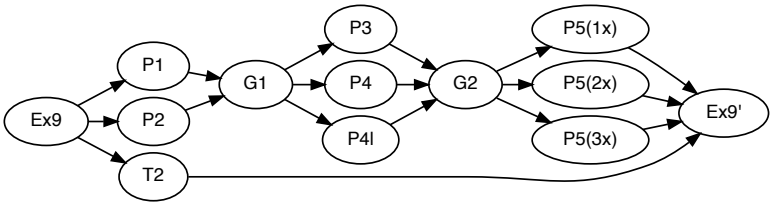

Figure 5.9: Process graph of *Ex9*.

many algorithms like Dijkstra's SPF [65] are based on edge weights, it is suggested to translate weights of vertices to edges. This may be done by *i)* splitting a vertex and adding an intermediate edge or by *ii)* applying the weight to an incoming/outgoing edge. The latter case results in a more compact graph and is used subsequently. While this modification allows the use of typical graph algorithms such as Dijkstra's SPF, the actual result of the graph analysis remains unchanged.

Following, the three different approaches for the application of graph weights are presented.

5.4.1 Separate Graphs

Separate graphs denote the first approach of criteria graphs. The process graph of *Ex9* depicted in Figure 5.9 is duplicated for each criterion of the chosen criteria set. Afterward, each criterion graph applies weights of the corresponding criterion to the related graph edges. This results in several graphs for a process, where each criterion is

allowed to use its own understanding of graph weights (e.g., value interpretation, value range, etc.). However, normalization of criteria values is useful to enhance comparability in many scenarios. For the criteria metrics, the abbreviations A ($\Rightarrow$ accuracy), C ($\Rightarrow$ cost), M ($\Rightarrow$ automation), P ($\Rightarrow$ privacy), R ($\Rightarrow$ resilience), and T ($\Rightarrow$ time) are used.

Figure 5.10 presents the criterion graph for communication resilience of process *Ex9*. Edge weights are based on connectivity characteristics where an edge weight $C_w^R \geq 1$ stands for resilient communication of the corresponding process element. The weights of Figure 5.10 illustrate that communication issues are only expected for the segment including *G1* $\rightarrow$ *P4* $\rightarrow$ *G2*. The repeating process segment including *P5* is not sharing the available communication resources and hence has the same edge weights for all possible repetition paths.

In this example, the criteria graphs for accuracy (Figure 5.11), cost (Figure 5.12), time (Figure 5.13), privacy (Figure 5.14) and automation (Figure 5.15) apply edge weights $C_w^x \in [0, 1]$. The weights do not reflect probabilities but give an indication of how good or bad a criterion is fulfilled. While for accuracy, privacy and automation values close to 1 are desired, cost and time target at low values close to 0. However, different values $C_w^x \in \mathbb{R}$ may be applied by other scenarios.

The accuracy graph in Figure 5.11 illustrates the low operation accuracy of *T2*. Since no tasks are following in the path of *T2*, the accuracy remains low for the whole path. The graph also illustrates that the repeating calls of *P5* improve the accuracy of *Ex9-c)* dramatically. However, the cost and time graphs in Figures 5.12 and 5.13 illustrate that the repetition in *Ex9-c)* increases the cost and the operating time of the process. Expenses for cost and time in the path of *T2* are minimal. Furthermore, operation times of the tasks of *Ex9-a)* and *Ex9-b)* differ. Considering the privacy of information processed by the different tasks of *Ex9*, *P1* shows the lowest privacy value $C_w^P = 0.3$ in the graph. *T2* is providing the highest privacy of all tasks with a value of $C_w^P = 1.0$. The same applies to the automation graph, describing the level of automation provided by tasks of *Ex9*. *P1* shows the lowest value of $C_w^M = 0.4$ and *T2* the highest value of $C_w^M = 1.0$ of the graph.

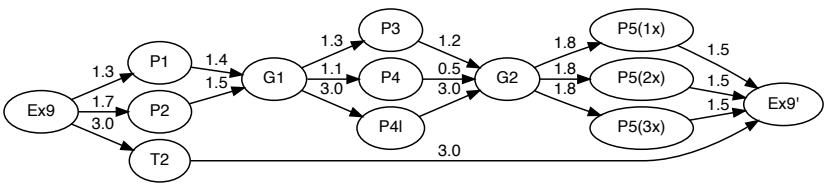

Figure 5.10: Resilience graph of *Ex9*, based on connectivity characteristic edge weights.

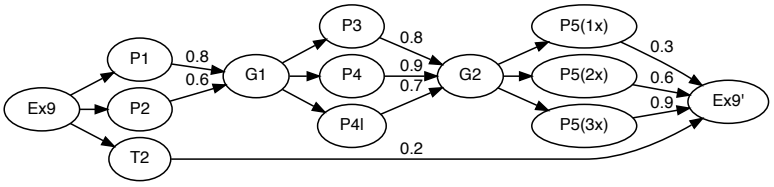

Figure 5.11: Accuracy graph of *Ex9*.

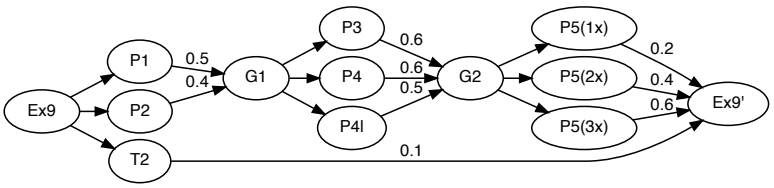

Figure 5.12: Cost graph of *Ex9*.

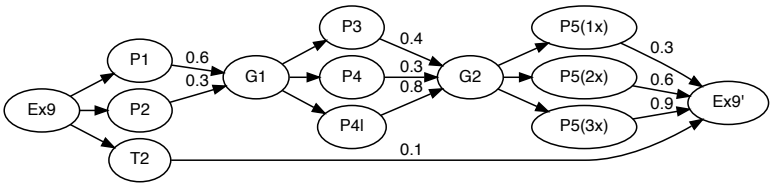

Figure 5.13: Time graph of *Ex9*.

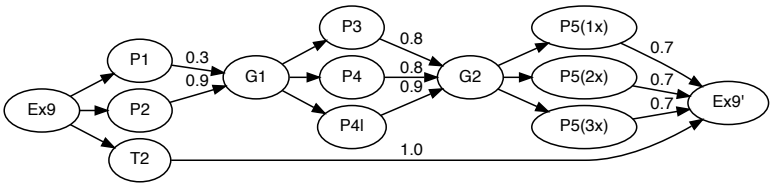

Figure 5.14: Privacy graph of *Ex9*.

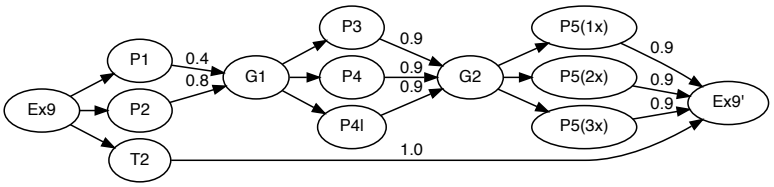

Figure 5.15: Automation graph of *Ex9*.

The duplication of graphs for different criteria allows utilization of well-known single-criterion graph algorithms with various implementations available for different programming languages (cf. subsequent section 5.5). Besides, there are no restrictions in adding weights to graph vertices and edges. Every criterion may choose to apply weights to a vertex, incoming or outgoing edge (cf. criteria graphs for resilience and accuracy of *Ex9*).

5.4.2 Joint Graphs

The second approach represents a frequently used technique in the area of multi-criteria optimization. Using the scalarization technique, the multi-criteria problem is reduced to a single-criterion problem (cf. section 2.3.2, p. 33f.). This is an appropriate concept for criteria that share a common understanding of edge weights, e.g., aiming at low edge weight values. If this is not the case, criteria values need to be harmonized. A subset or the whole set of criteria may be combined using a joint graph. This subsection illustrates the combination of a diverse criteria set to joint graphs. Techniques described in the literature are used for the merging of edge weights.

Applied on multi-criteria graphs, a scalarization technique is applied to merge the different criteria weights of a graph edge to a single, scalar edge weight. This allows utilization of single-criterion graph algorithms operating on scalar edge weights during the multi-criteria analysis, finding and deciding on a process path. The decision for one or another path is influenced by the result of the scalarization. Detailed elaborations about the graph analysis follow in the next section 5.5.

The literature introduces various scalarization techniques that can be used on multi-criteria graph weights. The *Weighted Sum Model (WSM)* is one of the most popular methods to reduce multiple values to a scalar value. It requires normalizing the corresponding criteria values, e.g., to graph edge weights $C_w^x \in \mathbb{R} | 0 \leq C_w^x \leq 1$. Criteria need to have the same understanding of weights, e.g., all aiming at low edge weights. Criteria may be weighted according to their importance for the process. During the scalarization, the different criteria values are summarized to a single value. Table 5.4 illustrates the scalarization procedure for the edge weights of cost and time and the resulting scalar weight. WSM is applied with 70 percent on cost and 30 percent on time. For instance, the joint weight for *P1* is calculated by $0.5 * 0.7 + 0.6 * 0.3 = 0.53$. Using the scalar weights, a joint graph for the criteria cost and time is created and depicted in Figure 5.16.

Instead of combining only the criteria of cost and time, all criteria may be combined in a scalar value to create a joint criteria graph of *Ex9*. This requires normalization of all criteria weights to a common value representation/understanding. Comparing the different criteria and their edge weights in Table 5.5 illustrates that most criteria values

Table 5.4: WSM combining normalized edge weights of cost (cf. Figure 5.12) and time (cf. Figure 5.13) to joint edge weights.

	Weight	*P1*	*P2*	*T2*	*P3*	*P4*	*P4l*	*P5 (1x)*	*P5 (2x)*	*P5 (3x)*
Cost	0.7	0.5	0.4	0.1	0.6	0.6	0.5	0.2	0.4	0.6
Time	0.3	0.6	0.3	0.1	0.4	0.3	0.8	0.3	0.6	0.9
Joint	1.0	0.53	0.37	0.10	0.54	0.51	0.59	0.23	0.46	0.69

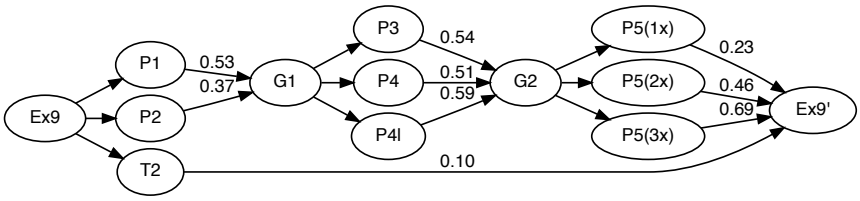

Figure 5.16: Joint criteria graph of *Ex9* for cost and time.

are defined within a range of 0 to 1. However, the resilience criterion has a range of 0 to 3. Other than cost and time, high values are beneficial for the criteria. Finally, all criteria except resilience are applied to the outgoing edge of a vertex. Harmonizing the resilience criterion to one value per outgoing edge may be realized by choosing the minimum of incoming and outgoing edge weights. Choosing the minimum edge weight is suggested since it avoids misleading statements about the resilience of communicating with a participant. Furthermore, rescaling the value to a range between 0 and 1 is advised. Harmonizing the values of cost and time can be done by inverting them.

Table 5.5: Preparation for the scalarization of all criteria edge weights of *Ex9*.

Criterion	**Range** [0,1]	**Objective** (low/high)	**Vertex edge** (in/out)	**Adjustment** (to create a common value understanding)
Resilience	x	high	in & out	pick min. value of in/out & rescale value
Accuracy	✓	high	out	-
Cost	✓	low	out	invert value
Time	✓	low	out	invert value
Privacy	✓	high	out	-
Automation	✓	high	out	-

Declaration: ✓⇒ within range x ⇒ not within range

Table 5.6: WPM combining normalized edge weights of resilience, accuracy, cost, and time to joint edge weights.

	Weight	*P1*	*P2*	*T2*	*P3*	*P4*	*P4l*	*P5 (1x)*	*P5 (2x)*	*P5 (3x)*
Resilience	0.50	0.43	0.50	1.00	0.40	0.17	1.00	0.50	0.50	0.50
Accuracy	0.25	0.8	0.6	0.2	0.8	0.9	0.7	0.3	0.6	0.9
Cost	0.20	0.5	0.6	0.9	0.4	0.4	0.5	0.8	0.6	0.4
Time	0.05	0.4	0.7	0.9	0.6	0.7	0.2	0.7	0.4	0.1
Joint	1.00	0.52	0.55	0.65	0.49	0.33	0.73	0.49	0.54	0.51

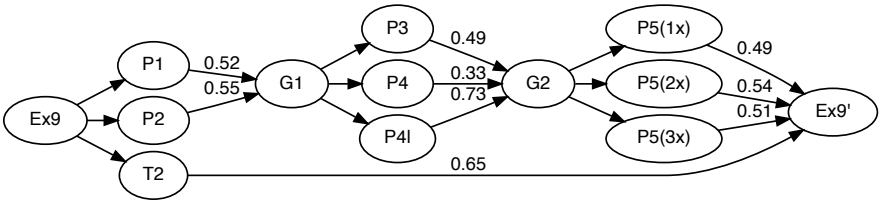

Figure 5.17: Joint criteria graph of *Ex9* for the optimization criteria resilience, accuracy, cost, and time.

Many alternatives to WSM are available. For instance, the *Weighted Product Model (WPM)* may be used for the scalarization of edge weights of *Ex9*. Instead of summarizing the values, WPM multiplies the values to a scalar. Table 5.6 presents the result of a WPM scalarization for the optimizing criteria of *Ex9*, namely resilience, accuracy, cost, and time. The criteria of privacy and automation have not been integrated since they are categorized as monitoring level type (cf. section 5.2). This results in the joint criteria graph for *Ex9* presented in Figure 5.17.

As illustrated, two or more criteria may be joint. Correspondingly, separate graphs and joint graphs may be used in combination. This is defined as merging only a subset of the criteria set to a joint graph while keeping separate graphs for the remaining criteria. The two concepts provide a high degree of flexibility, useful for the multi-criteria graph analysis presented in section 5.5.

5.4.3 Multi-Dimensional Graphs

The usage of algorithms specialized for the multi-criteria analysis may require a different type of representation for criteria graphs. Instead of having separate graphs or joint graphs, the third approach denotes the creation of a multi-dimensional graph. This graph has no singular edge weights, but arrays of weights for edges. This way, it is

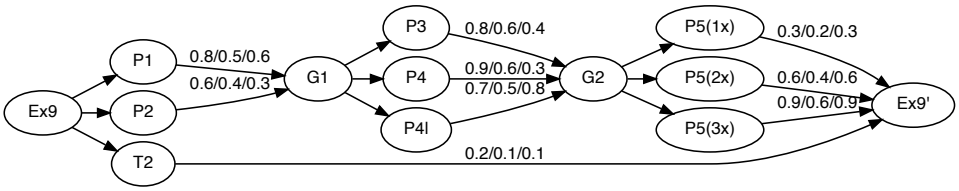

Figure 5.18: Multi-dimensional graph of *Ex9* with the criteria accuracy, cost, and time.

possible to remain having weights for every criterion while reducing the number of graphs to one. As a consequence, the principle of adding weights to vertices, incoming or outgoing edges may need to be unified across the set of criteria. Figure 5.18 depicts an example for a multi-dimensional graph for the criteria accuracy, cost, and time. The edges now include an array of three weight values for the chosen criteria, separated by slashes.

In this thesis, the concept of a multi-dimensional graphs is only relevant for using certain multi-criteria graph algorithms.

5.5 Multi-Criteria Graph Analysis

The objective of this section is to provide a versatile set of tools for the analysis of multi-criteria graphs. Due to the variety of scenarios, application domains, and possibly chosen criteria, different requirements for the multi-criteria graph analysis exist. This is addressed by providing three analysis approaches for different requirements: the *iteration-*, the *comparison-* and the *algorithm-based* analysis. As a fourth approach, a concept for the *scenario-based* customization and combination of the three approaches is outlined in section 5.5.4.

Compared to the single-criterion resilience analysis of chapter 4 (p. 65ff.), this analysis is more diverse by combining the demands of the different criteria into a common decision. This often requires comparing criteria with each other and applying distinct metrics, guaranteeing process operation as desired by domain experts. The combination of criteria can be a challenging task which is heavily influenced by the concrete scenario and application domain. Further on, this section provides examples for the usage and combination of multi-criteria metrics.

While this section uses graph algorithms described in the literature to find the best-suited process path, preparations are necessary to identify this path. A graph may need preparation in order to be applied with multi-criteria graph algorithms using the algorithm-based analysis approach. Since multi-criteria graph algorithms are often designed for specific use cases, customization capabilities regarding the treatment of different criteria may be limited (cf. section 2.3.2, p. 33f.). Publicly avail-

able implementations of these algorithms may be missing. Hence, the iteration- and comparison-based analysis approaches are introduced. Both approaches enable multi-criteria analyses using well-known and widely available graph algorithms, such as SPF and all-paths algorithms.

The section starts by introducing the iteration-based and comparison-based analysis approaches developed in this thesis. Following, the algorithm- and the scenario-based approaches are presented.

5.5.1 Iteration-based Analysis

The iteration-based analysis evaluates the graphs of the criteria set in a step-by-step manner. Classification and prioritization orders the criteria set as illustrated by Table 5.2 of section 5.2. By definition, criteria of the third importance level are ignored during analysis since they are categorized as non-optimizing monitoring criteria. Every iteration identifies graph paths that fulfill the minimum metrics defined for the current criterion. Unqualified edges that miss the defined metrics are removed from the graph. The following iteration takes over the graph adjustments on its criterion graph and continues by performing an analysis for its metrics. As illustrated in Figure 5.19, this procedure is repeated for every criterion until a remaining graph fulfilling the defined metric requirements is found. The analysis allows the application of different metrics for each criterion and to easily prioritize criteria against each other. The procedure guarantees that the resulting process paths meet the defined demands of the criteria and their metrics.

Identification and removal of unqualified edges may be realized by iterating through the graph. If beneficial, algorithms such as an all-paths graph search may be used for the graph iteration and to compare criteria metrics. Automation of the analysis procedure is possible by using an SPF or LPF graph algorithm on the final graph, finding the most appropriate process path. This is realized in Figure 5.20, presenting the result of an iteration-based analysis on the separate criteria graphs of *Ex9* for resilience, accuracy, cost, and time, illustrated in Figures 5.10 to 5.13. Starting with the resilience criteria, all edges with weights $C_w^R < 1.0$ have been removed. Accordingly, edges fulfilling the metrics $C_w^A < 0.4$ for accuracy, $C_w^C > 0.5$ for cost and $C_w^T > 0.8$ for time have been

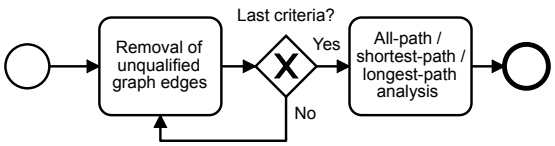

Figure 5.19: Iteration-based analysis procedure of criteria graphs (modeled in BPMN).

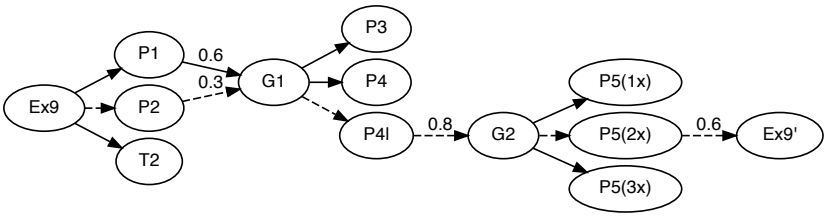

Figure 5.20: Final time criterion graph of *Ex9* showing an SPF search after an iteration-based analysis of resilience, accuracy, and cost.

removed in the following iterations. Finally, the optimal path is identified using an SPF algorithm on the remaining graph. The chosen path using *P2* → *P4l* → *P5(2x)* is depicted by a dashed line in Figure 5.20.

The iteration-based analysis is suitable for separate criteria graphs and scenarios including separate and joint criteria graphs. An example including mixed criteria graph types is process *Ex9* with separate graphs for the criteria resilience and accuracy, and a joint graph for cost and time (cf. Figure 5.16). If a scenario reduced the number of criteria graphs to a single graph during graph creation, there is no need to combine criteria in an iteration-based analysis anymore. However, there is no guarantee that an appropriate path can be found. If the scenario is not able to comply to the criteria demands, no path will be found. Identification of adequate criteria demands may require advanced knowledge about process behavior since minor changes may have a big impact on path choice. Using summarizing metrics such as the total cost C_t^C and total time C_t^T is challenging for the iteration-based analysis. Since these metrics are created by processing numerous weights, it is not possible to identify them when investigating single graph weights. However, summarizing metrics may be used in finding the optimal path in the remaining graph of an iteration-based analysis.

5.5.2 Comparison-based Analysis

The comparison-based analysis for multi-criteria graphs starts by evaluating all graphs using an all-paths graph algorithm. Identified paths are compared by chosen metrics representing the demands of the different criteria to find the most appropriate path available. While this approach represents a broader, more complete type of graph analysis, it requires a higher level of manual interaction in comparing and choosing process paths. A useful procedure is to filter process paths by minimum or maximum criteria metrics first (e.g., C_l^x and C_h^x) and to compare the remaining paths afterward. Figure 5.21 illustrates the comparison-based analysis procedure.

The outcome of an all-paths algorithm applied on process *Ex9* is listed in Table 5.7. A total of 19 different paths has been identified, along with chosen metrics for

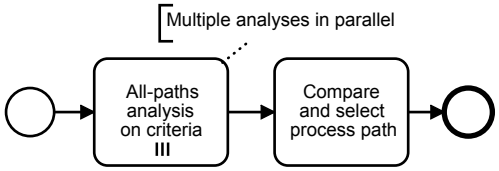

Figure 5.21: Comparison-based analysis procedure of criteria graphs (modeled in BPMN).

all levels of importance (optimization and monitoring criteria). The metrics differ for each criterion and include the lowest path weight C_l^x, average path weight C_a^x and summarized total path weight C_t^x. An additional column states whether or not the process path is Pareto-optimal or not.

Finding *Pareto-optimal solutions* is a common strategy for multi-criteria optimization (cf. section 2.3.2, p. 33f.). Regarding listed process paths in Table 5.7, Pareto evaluation is based on the first appearing metric for every criterion (e.g., C_l^R instead of C_t^R). Twelve Pareto-optimal paths have been identified by comparing all paths with each other. However, seven of the Pareto-optimal paths seem to be inappropriate compared to other paths. Path numbers 6, 13, 14, and 15 are Pareto-optimal but non-resilient due to their lowest path weight $C_l^R = 0.5$. Path numbers 10, 13, 16, and 19 are resilient but have a poor accuracy which makes them an unlikely choice as suitable process configurations. Path numbers 3, 11, 12, 17, and 18 are Pareto-optimal configurations which may be considered for selection due to their solid criteria metric values.

Table 5.8 lists selected paths for process *Ex9*, filtered by excluding all paths not fulfilling the metrics $C_l^R \geq 1.0$ for resilience and $C_l^A \geq 0.4$ for accuracy. Besides, the evaluating criteria metrics have been limited to C_l^R, C_l^A, C_t^C and C_t^T. Especially for cost and time, the total path weights meet the demands of these criteria better, indicating the total cost and total time required by the corresponding process path.

Filtering the possible process paths by lowest edge weights for resilience and accuracy not only reduces the number of paths, but also guarantees to comply with the minimum requirements for the aforementioned criteria. A domain expert may compare and choose one of the remaining paths of Table 5.8, e.g., by reducing cost and time efforts required for process operation. Graphical forms of representation are often beneficial to compare the available paths. For instance, Figures 5.22 to 5.25 illustrate the different paths and the belonging criteria in radar charts. The Pareto-optimal path number 3 would be a reasonable choice with high accuracy. However, the Pareto-optimal paths 11 and 17 may be a more appropriate choice due to their reduction of cost and time efforts.

Table 5.7: List of all paths for a comparison-based analysis of criteria graphs for *Ex9*.

#	**Path variation**	**Resilience**		**Accuracy**		**Cost**		**Time**		**Privacy**		**Automation**		**Pareto**
	Ex9 → ... → Ex9'	C_l^R	C_t^R	C_l^A	C_a^A	C_t^C	C_a^C	C_t^T	C_a^T	C_l^P	C_a^P	C_l^M	C_a^M	
1.	*P1 → P3 → P5(1x)*	1.2	8.5	0.3	0.6	1.3	0.4	1.3	0.4	0.3	0.6	0.4	0.7	-
2.	*P1 → P3 → P5(2x)*	1.2	8.5	0.6	0.7	1.5	0.5	1.6	0.5	0.3	0.6	0.4	0.7	-
3.	*P1 → P3 → P5(3x)*	1.2	8.5	0.8	0.8	1.7	0.6	1.9	0.6	0.3	0.6	0.4	0.7	✓
4.	*P1 → P4 → P5(1x)*	0.5	7.6	0.3	0.7	1.3	0.4	1.2	0.4	0.3	0.6	0.4	0.7	-
5.	*P1 → P4 → P5(2x)*	0.5	7.6	0.6	0.8	1.5	0.5	1.5	0.5	0.3	0.6	0.4	0.7	-
6.	*P1 → P4 → P5(3x)*	0.5	7.6	0.8	0.9	1.7	0.6	1.8	0.6	0.3	0.6	0.4	0.7	✓
7.	*P1 → P4l → P5(1x)*	1.3	12	0.3	0.6	1.2	0.4	1.7	0.6	0.3	0.6	0.4	0.7	-
8.	*P1 → P4l → P5(2x)*	1.3	12	0.6	0.7	1.4	0.5	2.0	0.7	0.3	0.6	0.4	0.7	-
9.	*P1 → P4l → P5(3x)*	1.3	12	0.7	0.8	1.6	0.5	2.3	0.8	0.3	0.6	0.4	0.7	-
10.	*P2 → P3 → P5(1x)*	1.2	9.0	0.3	0.6	1.2	0.4	1.0	0.3	0.7	0.8	0.8	0.9	✓
11.	*P2 → P3 → P5(2x)*	1.2	9.0	0.6	0.7	1.4	0.5	1.3	0.4	0.7	0.8	0.8	0.9	✓
12.	*P2 → P3 → P5(3x)*	1.2	9.0	0.6	0.8	1.6	0.5	1.6	0.5	0.7	0.8	0.8	0.9	✓
13.	*P2 → P4 → P5(1x)*	0.5	8.1	0.3	0.6	1.2	0.4	0.9	0.3	0.7	0.8	0.8	0.9	✓
14.	*P2 → P4 → P5(2x)*	0.5	8.1	0.6	0.7	1.4	0.5	1.2	0.4	0.7	0.8	0.8	0.9	✓
15.	*P2 → P4 → P5(3x)*	0.5	8.1	0.6	0.8	1.6	0.5	1.5	0.5	0.7	0.8	0.8	0.9	✓
16.	*P2 → P4l → P5(1x)*	1.5	12.5	0.3	0.5	1.1	0.4	1.4	0.5	0.7	0.8	0.8	0.9	✓
17.	*P2 → P4l → P5(2x)*	1.5	12.5	0.6	0.6	1.3	0.4	1.7	0.6	0.7	0.8	0.8	0.9	✓
18.	*P2 → P4l → P5(3x)*	1.5	12.5	0.6	0.7	1.5	0.5	2.0	0.7	0.7	0.8	0.8	0.9	✓
19.	*T2*	3.0	6.0	0.2	0.2	0.1	0.1	0.1	0.1	1.0	1.0	1.0	1.0	✓

Declaration: ✓⇒ Pareto-optimal path - ⇒ no Pareto-optimal path

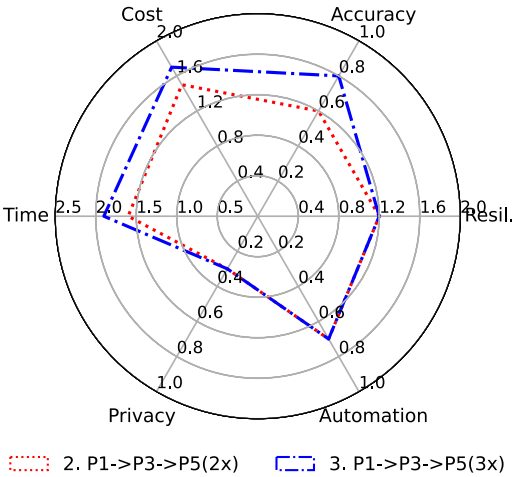

Figure 5.22: Radar chart of *Ex9* for results including vertices *P1, P3*.

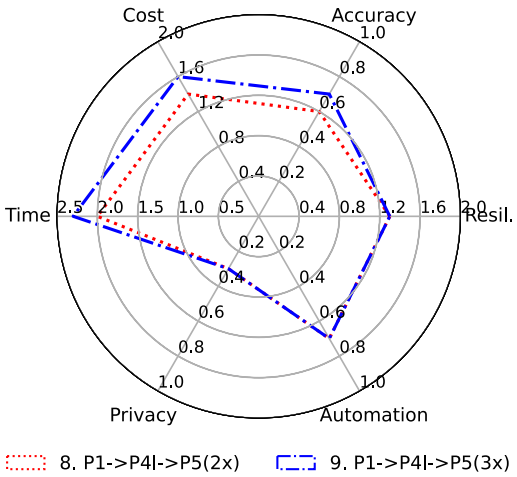

Figure 5.23: Radar chart of *Ex9* for results including vertices *P1, P4l*.

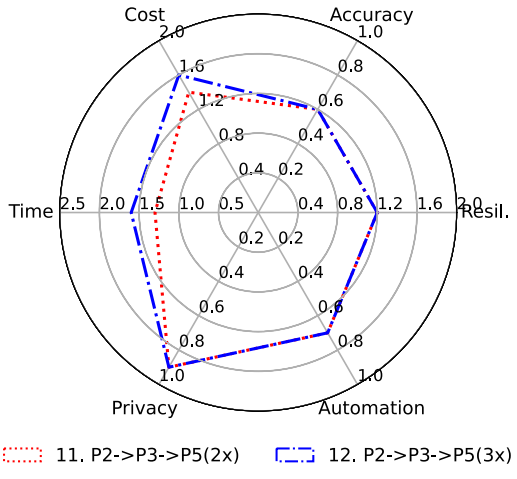

Figure 5.24: Radar chart of *Ex9* for results including vertices *P2, P3*.

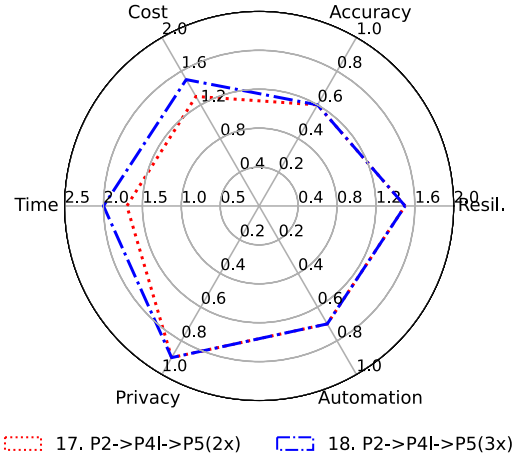

Figure 5.25: Radar chart of *Ex9* for results including vertices *P2, P4l*.

Table 5.8: Selected paths of *Ex9* used in a comparison-based analysis of criteria graphs.

#	**Path variation**	**Resilience**	**Accuracy**	**Cost**	**Time**	**Privacy**	**Automation**	**Pareto**
	Ex9 → ... → Ex9'	C_l^R	C_l^A	C_t^C	C_t^T	C_l^P	C_a^M	
2.	*P1 → P3 → P5(2x)*	1.2	0.6	1.5	1.6	0.3	0.7	-
3.	*P1 → P3 → P5(3x)*	1.2	0.8	1.7	1.9	0.3	0.7	✓
8.	*P1 → P4l → P5(2x)*	1.3	0.6	1.4	2.0	0.3	0.7	-
9.	*P1 → P4l → P5(3x)*	1.3	0.7	1.6	2.3	0.3	0.7	-
11.	*P2 → P3 → P5(2x)*	1.2	0.6	1.4	1.3	0.7	0.9	✓
12.	*P2 → P3 → P5(3x)*	1.2	0.6	1.6	1.6	0.7	0.9	✓
17.	*P2 → P4l → P5(2x)*	1.5	0.6	1.3	1.7	0.7	0.9	✓
18.	*P2 → P4l → P5(3x)*	1.5	0.6	1.5	2.0	0.7	0.9	✓

Declaration: ✓⇒ Pareto-optimal path - ⇒ no Pareto-optimal path

5.5.3 Algorithm-based Analysis

An outline of additional techniques for the analysis of multi-criteria graphs is provided subsequently. The presented approaches are often driven by specialized graph algorithms, integrating techniques to address the finding of a graph path according to the criteria set. The objective of this subsection is to illustrate relevant aspects when applying single- and multi-criteria graph algorithms.

The principle of scalarization is used in section 5.4.2 to create joint graphs of multiple criteria. A joint graph can be used in conjunction with graph-based search algorithms, operating on scalar edge weights. Using all-paths algorithms, metrics for different paths can be identified. Application of SPF and LPF algorithms may allow to automatically identify the most suitable path based on the joint edge weights. An LPF analysis has been performed on the joint graph of process *Ex9* in Figure 5.26, depicting the chosen path by a dashed line. In general, the same set of tools as for the analysis of resilience graphs in section 4.3 (p. 85ff.) is suitable for joint graphs.

Another possibility for the analysis of multi-criteria graphs is to keep the set of criteria-based weights for every edge and to apply graph algorithms designed for a multi-criteria analysis. As outlined in section 2.3.2 (p. 33f.), different approaches are described in the literature. Some of the algorithms integrate the scalarization of edge weights as part of the graph algorithm. Depending on the concrete algorithm, different scalarization techniques may be used. Other algorithms keep separate criteria weights

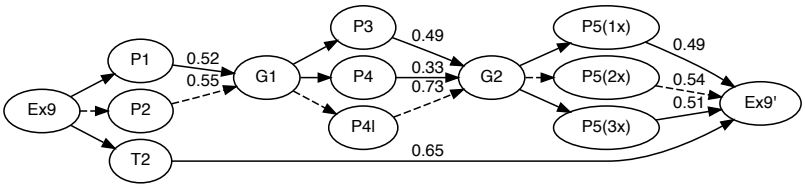

Figure 5.26: LPF analysis on joint graph of *Ex9*, longest path as dashed line.

and analyze the graph by finding all Pareto-optimal paths. Depending on the graph algorithm, multi-dimension graphs may be required.

The impacts of multi-criteria graph algorithms on the process path choice are mainly comparable to the effects of joint graphs. Algorithms using scalarization may not be able to apply metrics bound to a certain criterion anymore. For instance, it may not be able to identify non-resilient or unqualified edges anymore, which may be part of the chosen process path. Algorithms identifying Pareto-optimal paths require a subsequent analysis to select a path from the Pareto-optimal path set. However, the most suitable path of a process may not be a Pareto-optimal path.

The suitability of multi-criteria graph algorithms often depends on the application domain since most algorithms represent solutions for problem statements of specific scenarios. Many algorithms are based on research papers; the source code may be missing. Evaluations of algorithms are often based on simulations or prototypes; experiences from widespread usage in the real world are missing.

5.5.4 Scenario-based Analysis

The last type of multi-criteria analysis is not an approach on its own, but a recommendation for the scenario-based usage and combination of analysis methods provided by the iteration-, comparison- and algorithm-based analysis approaches. Each of the three approaches has unique characteristics, resulting in advantages and disadvantages. The characteristics are outlined in Table 5.9. Explanations and conclusions follow subsequently.

Since the requirements of scenarios and application domains are diverse, there is no universally applicable recommendation. The comparison-based analysis provides the most insights regarding criteria metrics and possible paths. Selection of a path is not automatable directly by all-paths algorithms, but by comparing the paths with selected metrics.

The iteration- and algorithm-based approaches can reduce efforts by automating the analysis but may lack the required details about certain criteria metrics. For instance, an iteration-based analysis analyzes path-based metrics only on the final, remaining

Table 5.9: Characteristics of different approaches for the analysis of multi-criteria graphs.

Approach	**Metrics** (path-based)	**Metrics** (edge-based)	**Demands** (ensured)	**Auto-matable**	**Algorithm** (example)
Iteration	✓[1]	✓	✓	✓	Iteration/SPF/LPF
Comparison	✓	✓	✓	x[3]	all-paths
Algorithm (scalar-based)	✓[2]	✓[2]	x	✓	specialized
Algorithm (Pareto-based)	✓	✓	x	✓[4]	specialized

Declaration:
x unfulfilled ✓fulfilled ✓[1] on the final graph ✓[2] only on joint criteria
x[3] not by the graph algorithm itself, but by ranking paths using criteria metrics
✓[4] list of Pareto-optimal paths as result

graph after all iterations are completed. Also, scalar-based algorithms are unable to identify distinct criteria metrics since the criteria have been joined. Finally, Pareto-based algorithms may automatically find sets of Pareto-optimal paths, but are unable to decide on one of the paths.

A recommendation applicable to many scenarios is to comprehensively analyze the behavior of a process at design time to get familiar with the usefulness and the expectations of metrics. Afterward, the analysis effort in a now-familiar process behavior may be reduced for runtime execution. A combination of a comparison-based analysis at design time and an iteration-based analysis at runtime may be appropriate. Alternatively, a joint graph may fit for the runtime analysis.

An example for a scenario-based multi-criteria analysis is provided as part of the evaluation in section 7.1 (p. 140ff.).

Summary

Many business processes include multiple criteria to be respected for optimal operation. This chapter addresses challenge 3 of this thesis, the multi-criteria process operation. For this purpose, different approaches for the multi-criteria analysis of processes are introduced.

Different criteria influence the outcome of a multi-criteria analysis. Every criterion and its relevance for the process should be clearly defined. For most scenarios, it is beneficial to categorize and prioritize the criteria against each other, e.g., by using the introduced importance levels.

Selecting the most appropriate metrics often results in having distinct metrics for different criteria. For instance, the lowest path weight C_w^x seems to represent a solid metric to measure resilience and accuracy in a graph. However, the total path weight C_t^x is preferable when measuring operational cost and time of process paths. Selecting multiple metrics to compare them during the graph analysis is advised for criteria not showing a favored metric.

The process-to-graph translation is comparable to the translation of processes to resilience graphs. However, process segments including repetitions require additional attention. Here, some criteria may require adding repetition-based edge weights while other criteria apply the same weights to every repetition. Also, simplifying segments not using communication is no longer possible. Many criteria like operation time are activity-driven, different paths require consideration even though they avoid communication with other participants.

Application of criteria weights may be realized by using separate, joint, or multi-dimensional graphs. Analyzing separate graphs in an iteration-based manner allows to keep track of criteria-driven metrics. For instance, it is ensured that the minimum demands of criteria will be met by the resulting process path. However, there is not always a suitable path fulfilling the criteria requirements. Joint graphs enable weighted prioritization of criteria against each other and simplify the analyzing effort. Unfortunately, joint graphs no longer allow inspecting criteria-specific metrics (e.g., lowest path weight of a criterion). If available, graph algorithms for the multi-criteria analysis may be suitable. Some paths of a Pareto-optimal path set may be considerable for selection. However, it remains uncertain if Pareto-optimal paths are reasonable choices for every scenario. A comprehensive analysis of the suitability of Pareto paths is advised. In conclusion, the recommended type of criteria graph analysis depends on the concrete scenario and application domain.

CHAPTER 6

RESILIENT PROCESS EXECUTION

The previous chapters cover the modeling and analysis of business processes in unreliable communication environments. This chapter focuses on the execution and operation of modeled processes. The steps of designing and analyzing process models take a major role in the resilient operation of processes. Many issues in terms of communication resilience and non-optimal process operation can be identified and prevented before starting a process. However, connectivity conditions may differ in the real world compared to the conditions that were considered at design time. Also, many processes show dynamic behaviors which may lead to deviations from the planned execution. For instance, the mobility of process participants may have changed due to modifications in process procedures, resulting in the unavailability of the participants. Other participants may become dynamically available offering equivalent functionality compared to the originally planned participants' functionality. Hence, ensuring resilient process operation requires to constantly monitor and adapt the process and its parameters to environmental influences.

rBPMN introduces concepts for choosing the best-suited alternative that has connectivity, for the dynamic integration of participants at runtime and for moving and executing functionality locally in cases of no connectivity. Although these resilience strategies are part of the *rBPMN* metamodel, there are no technical specifications on which technologies should be used and how the strategies should be implemented. Strategies for the execution of resilient process models are missing.

The focus of this chapter is twofold: At first, requirements for the resilient execution of business processes are determined. Strategies satisfying the requirements are elaborated by combining state-of-the-art paradigms and technologies with newly developed concepts and procedures. Secondly, the flexibility of *rBPMN's* decision-making is illustrated by presenting a graph-based and a WSM-based approach for the process analysis at runtime. In this context, aspects of integrating process dynamics into decision-making are discussed.

6.1 Resilience Strategies for Process Execution

The requirements for the execution of a resilient process modeled in *rBPMN* are identified subsequently.

The start of a process requires an initial set-up configuration to specify process variables and communication parameters. In unreliable communication environments, infrastructure-based (e.g., cellular networks, WiFi in access-point mode) and infrastructure-free (e.g., WiFi in ad-hoc mode) technologies are frequently combined to hybrid networks (cf. section 2.1.2, p. 11f.). Configuration settings are required to operate and access the networks. This results in the following concluded requirement:

Req. E1: Ability to set-up the initial process and communication configuration before runtime.

rBPMN introduces process elements offering movable functionality to be executed locally at other participants. Hence, a mechanism for moving functionality to interested participants is needed. It should not matter whether the participants belong to a common or different organization. This leads to the following requirement:

Req. E2: Ability to move functionality between participants of different organizations.

A running process needs to recognize neighboring participants dynamically, since they may be part of the same process model. Also, a process may identify and use offered functionality of neighbors to adapt its operation to the best-suited alternatives available. This concludes to the following requirements:

Req. E3: Ability to discover participants dynamically at runtime.

Req. E4: Ability to identify and use functionality offered by other participants at runtime.

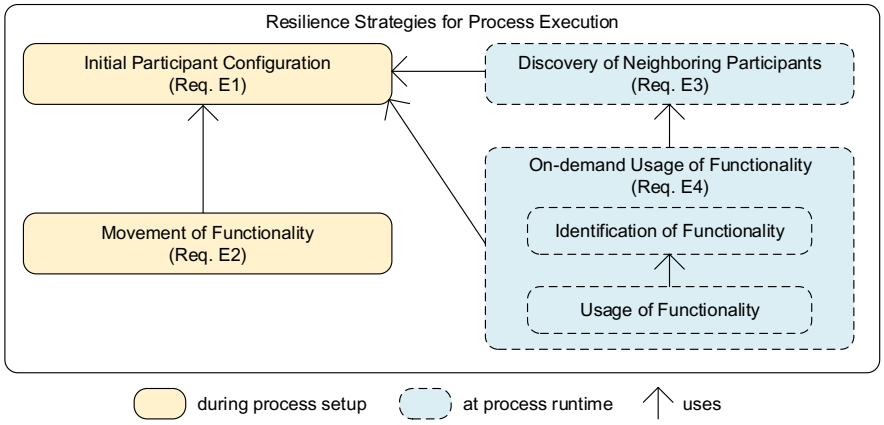

Figure 6.1: Resilience Strategies for Process Execution and their relations.

Due to the dynamics taking place in many processes, the decision-making procedure needs to integrate updated connectivity and service information into its decisions at runtime. This issue is addressed in section 6.2.

Based on the identified requirements, four resilience strategies for process execution are elaborated in the following subsections. Figure 6.1 illustrates the strategies including their relations with each other and to the requirements. This subsection applies state-of-the-art paradigms and exemplary technologies to be used for implementation.

6.1.1 Initial Participant Configuration

The strategy for the initial process and network configuration (*Req. E1*) introduced by this thesis includes a management entity (*MGMT*) located in the cloud, providing configurations to all participants. *MGMT* is contacted by every participant to retrieve their configuration, including process variables and communication settings. Since network addressing may change rapidly and a participant may be part of multiple networks at the same time, a *participant ID* is used for identification. Participants transfer their offered, movable functionality to *MGMT* for distribution to other participants. This way, the functionality may be used locally at a participant in the case of connectivity issues (cf. section 3.3, p. 47ff.). Figure 6.2 summarizes the initial configuration sequence, supposed to take place before process execution when reliable connectivity is available.

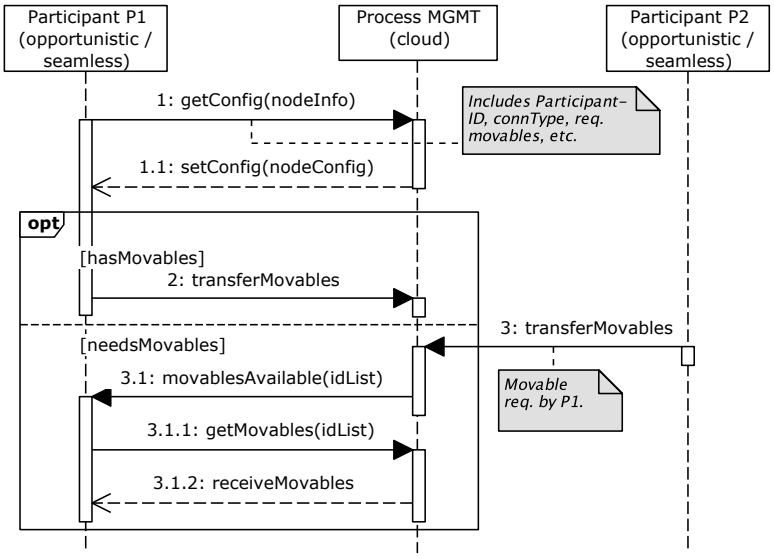

Figure 6.2: Participant configuration and transfer of movable functionality for local execution during the initial set-up sequence of a process.

The configuration messages contain relevant instructions to prepare the participants for the process execution. Many use cases will find three configuration categories useful:

i) Process-related instructions such as process variables/parameters, offered and consumed functionality, IDs of all process participants.

ii) Network-related configuration settings such as addressing, naming, user credentials, routing settings.

iii) Service-related information such as service and functionality IDs, service addressing of seamlessly connected participants.

A JSON-formatted example is included in the source code of the proof-of-concept, part of appendix A (p. 241ff.).

6.1.2 Movement of Functionality

A local backup of (limited) functionality ensures process operation for participants even if no connectivity is available (*Req. E2*). Functionality needs to be movable in the sense of mobile-code / code-on-demand [73]. Depending on the software platform and BPMN runtime engine used in a scenario, several approaches are applicable to design movable process parts:

Engine-bound process modules If all participants agree on the same BPMN runtime engine, process parts may be exchanged directly as engine modules. A common format for process modules are Java-archives (.jar) since many runtime engines are based on Java. It is important to add all required libraries to the archive, as they need to be present at other participants.

Microservices Realizing functionality as a microservice allows convenient functionality movement across participants. All dependencies and data artifacts required for execution are part of the microservice. BPMN runtime engines can be integrated into a microservice, eliminating the need for a local runtime instance at other participants. However, required software platforms such as Java still need to be present on the remote participants.

Container virtualization Mobile-code may be realized by container virtualization techniques like *docker* [48] and *rkt* [146], especially if different software platforms and BPMN runtime engines are applied by the participants. The provided functionality is encapsulated within a container with all of its dependencies and can be executed locally by other participants. All participants need to run the required container technology.

For seamless integration into the process, the participants' service registry may be used to register and dynamically access the functionality within the running process. Service registries are part of the service discovery, outlined in section 6.1.4. To ensure the availability of local components, the initial configuration sequence moves required functionality to the corresponding participants before process runtime (cf. Figure 6.2).

6.1.3 Discovery of Neighboring Participants

An essential part of process operation is to recognize neighboring participants (*Req. E3*). Two participants are considered as neighbors if they are in close proximity and can communicate with each other using an ad-hoc communication technology. Cloud-connectivity of these participants may be opportunistic or non-existing. Participants often show a high level of mobility. Neighboring participants may be part of the process model serving as functionality providers to run and optimize a process.

The proposed approach realizes neighbor detection by using routing algorithms for MANETs. Proactive routing algorithms such as DSDV and OLSR detect neighbor nodes by picking up periodic *hello* broadcast messages emitted by every node of the network (cf. section 2.1.3, p. 12f.). The frequency of the *hello* broadcast messages determines the speed of neighbor detection and should ensure the detection in a reasonable amount of time, depending on the applied use case. The frequency can be

aligned for every participant separately and is part of the initial configuration sequence (cf. section 6.1.1).

Figure 6.3 illustrates the neighbor discovery mechanism. When a neighbor is detected by a proactive routing algorithm, its IP address is entered into a custom neighbor participant table. After detection, the participant is identified by requesting node information on a fixed port (e.g., port 9876). Node information includes a *participant ID* and may state the port number of the local service registry if the participant offers functionality. A custom table is required to group participants by their IDs, since participants part of multiple networks may have diverse IP addressing information. Also, such a custom table helps to maintain functionality information offered by neighbors. The offering of functionality will be discussed in the following subsection.

When a participant exits the communication range of the MANET, *hello* broadcasts are no longer received by the applied routing algorithm. The participant is removed after a timeout period from the neighbor table.

The detection of participants that are connected by cellular communication works differently. Whenever connected, the participants provide their (updated) addressing information to *MGMT*, which is placed in the cloud. Depending on the cellular network provider, the use of a Virtual Private Network (VPN) may be required to ensure that the mobile participants are not only able to access the cloud but to be accessed from other participants. *MGMT* serves as an addressing/networking gateway to communicate

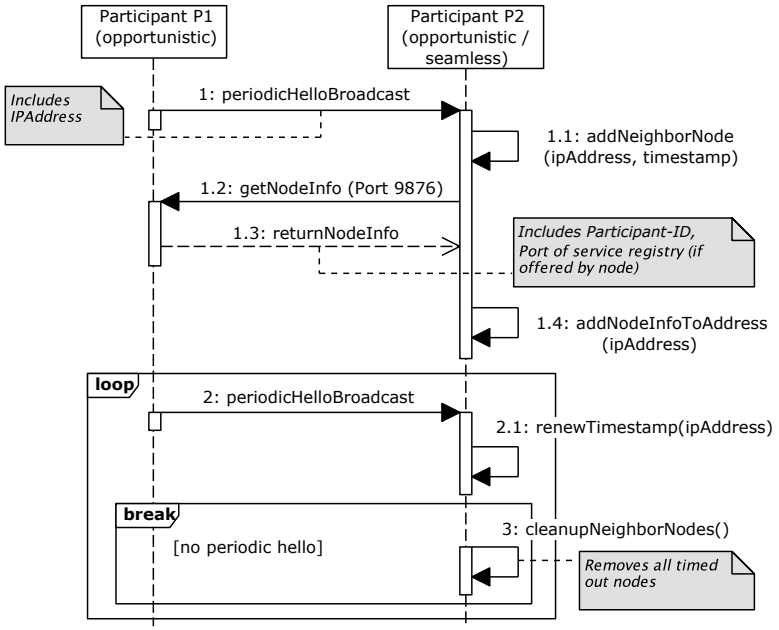

Figure 6.3: Discovery and identification of neighboring participants.

with the mobile participants connected by cellular networks. This can be realized by running a domain name service at *MGMT*. If a VPN is required, *MGMT* may operate as a VPN server connecting all cellular participants.

Mobile participants often combine different communication technologies. If participants include cellular communication and ad-hoc communication technologies, they may serve as connecting elements for ad-hoc participants missing a connection to the cloud. By publishing this routing information to *MGMT*, all other participants may communicate with ad-hoc participants using the connecting participants.

Seamlessly connected participants that reside in the cloud keep their address settings during process runtime. While their addressing information is placed at *MGMT*, it is also part of the initial configuration sequence. This way, mobile participants may communicate without the support of *MGMT* with seamlessly connected cloud participants.

While the described discovery of neighboring participants lists currently available neighbors, future version may provide predictions about communication opportunities. Information of routing algorithms using temporal context, geographic locations, and social graphs may be beneficial. In addition, information about upcoming communication opportunities may be gathered from algorithms designed for DTNs (cf. section 2.1.3, p. 12ff.).

6.1.4 On-demand Usage of Functionality

After discovering neighboring participants, mechanisms for the identification and usage of offered functionality (*Req. E4*) are presented subsequently.

Identification of Functionality

The integration of functionality provided by other participants into a process requires identifying the functionality at first. Section 3.3 introduced an ontology (cf. Figure 3.3, p. 49) to guide the offering and usage of functionality. The ontology is describing a common rule set for all functionality providers, simplifying the process of identification and usage across different participants and organizations.

According to the ontology of Figure 3.3, functionality is part of services and can be identified by IDs and descriptions. Service information is gathered by service registries, which offer the service information to requesting participants. Using the unique IDs for functionalities and services, a participant can identify if the type of functionality is fitting the required one, and what service (or also what participant) is providing it. Participants may request lists of services from the service registries as well as detailed metadata describing services.

A central service registry is located at *MGMT*. Besides, ensuring service usage in unreliable environments requires a mechanism for opportunistically connected participants. Hence, every opportunistic participant provides its own service registry, offering service information and usage for requesting neighbors.

Usage of Functionality

Using functionality requires a description of interfaces with input and output parameters. While in the days of web services *WSDL* [37] and *WADL* [170] have been used for interface descriptions, application of technologies like *OpenAPI* [131] and *RAML* [143] is widespread in the area of microservices. Organizations may standardize interfaces to support interoperability of implementations. For certain use cases, guidance provided by the *Hypermedia as the Engine of Application State (HATEOAS)* principle of *Representational State Transfer (REST)* may be appropriate (cf. [69]). With HATEOAS, the service is offering links for functionality currently available based on service state information.

Due to the opportunistic nature of unreliable communication environments, the following guidelines have to be followed independently of a concrete implementation:

1) Services describe their functionality using metadata.

2) Services are registered in a service registry.

3) Participants with opportunistic connectivity operate their own service registry.

4) Every participant runs an information service following a well-defined interface for identification needs / as an entry point for the usage of service functionality.

The solution strategy to dynamically identify and use functionality is illustrated by Figure 6.4. A neighboring participant with opportunistic connectivity is asked for a service list on its service registry port. The returning services are queried for service metadata to identify the service categories, IDs, and interfaces. In the case of HATEOAS, functionality links are provided based on the current state. The functionality may be used by calling the appropriate service interfaces.

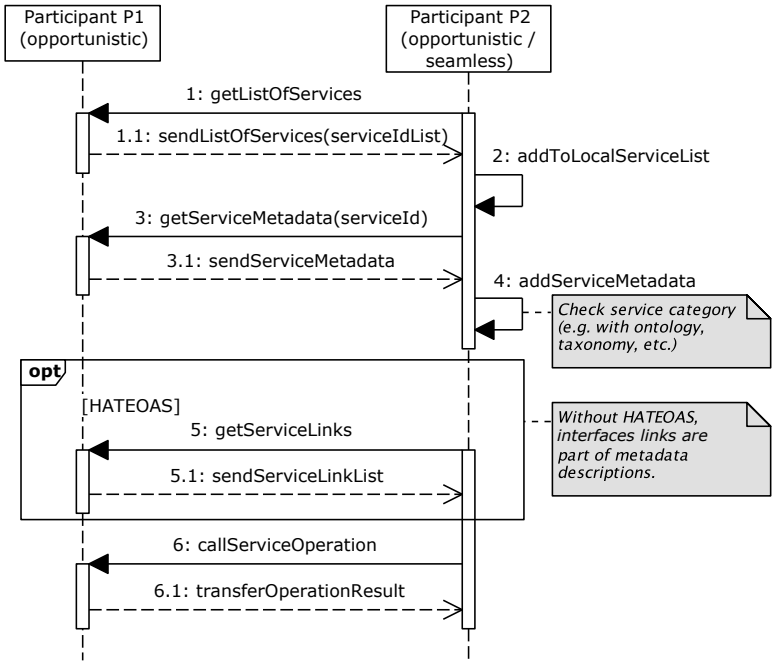

Figure 6.4: Dynamic identification and usage of service functionality.

6.2 Process Optimization at Runtime

The resilience strategies for process modeling of *rBPMN* allow the definition and verification of resilient process models (cf. chapters 3 and 4, p. 41ff.). Further on, optimal process operation regarding a diverse set of criteria may be analyzed (cf. chapter 5, p. 95ff.). While the graph-based analysis approaches for decision-making may be employed at design and at runtime, integration of alternative concepts for decision-making is also possible. This section outlines the use of a graph-based and an alternative WSM-based decision-making at runtime. Since processes may change dynamically at runtime, the integration of dynamics into the decision-making on a process path is discussed.

6.2.1 Continuous Graph- and Decision-Updating

Chapters 4 and 5 (p. 65ff.) illustrate the decision-making for the most suitable process path based on DAGs. These graphs represent the foundation of decisions. Ensuring resilient operation of a process requires to constantly update the graph structure and the corresponding weights according to the environmental conditions. The subsequent paragraphs illustrate how dynamics regarding planned and actual connectivity as well as regarding participant- and process behavior can be addressed by *rBPMN*. Afterward, effects of process dynamics on the graph analysis are discussed.

Planned and Actual Connectivity

Typical dynamics faced by processes executed in unreliable communication environments are deviations in planned and actual connectivity of participants. Depending on the experience with a process in a concrete scenario, the quality of connectivity estimations for the resilience analysis at design time differs. The connectivity of certain participants may be worse or better than expected. Hence, corresponding edge weights need to be constantly updated with the runtime connectivity of participants. Connectivity information may be gathered by monitoring parameters of the participants' network connections.

The signal quality, signal-to-noise ratio and the configured connection speed may indicate the current bandwidth of the connection. Periodical speed tests should be used with caution: While they help to identify the available bandwidth of a connection, connectivity resources are consumed during the test period. This does not only affect the testing participant but all participants in the same network segment sharing the communication resources. For instance, if the bandwidth is consumed by a participant using WiFi, all other participants connected to the same access point will not be able to consume that bandwidth. Besides, a speed test is only an extract of the current communication conditions. Due to other communication activities in the network segment, the achievable bandwidth may significantly vary at a later time.

Connectivity information may be shared by services across all participants. Following the principle of providing functionality in section 6.1.4, other participants may request connectivity information of a participant to update their graph weights.

Dynamics in the Participant- and Process-Behavior

Some scenarios show extensive dynamics in the mobility of participants due to their role in the process. For instance, many participants moving goods across the scenario are constantly meeting other participants and may interact with them. However, planning interaction at design time may be challenging if the meetup is a side effect of the participants' main objective.

Similarly, the actual process behavior is influenced by parameters captured by process variables. How many times a repetition part of a process model needs to be passed usually depends on the process status, captured by variables and processed in modeled decision points.

OppPriorityFlows of *rBPMN* allow modeling a prioritized set of alternatives for communication with other participants. The available alternative with the highest priority is chosen dynamically at runtime. With *OppDecisionFlows*, it is possible to avoid fixed priorities and to apply decisions based on comparable features. Both modeling

concepts address expected dynamics that may occur in a process. Besides, two options exist to cope with unplanned and unexpected dynamics: *i) OppDynTasks* of *rBPMN* enable the integration of unplanned participants that offer suitable functionality dynamically at runtime and *ii)* functionality may be moved and executed locally.

All structural process changes require immediate integration into the process graph. This may happen directly before the process start, if the required information is available during process initiation. Otherwise, integration has to take place at process runtime.

Dynamic Graph Analysis

The previous two subsections illustrate the need for rapid re-runs of the graph analysis as the method of decision-making. Whenever the process graph is updated with new edge weights or repetition information, appearing or disappearing participants, a subsequent graph analysis needs to re-evaluate the process objectives. The result may confirm the current chosen path, report a more suitable path option or indicate non-resilient operation in process segments ahead. The process may be adapted immediately to a more resilient or optimal operation. Especially in situations where the original path has become non-resilient, an early discovery allows avoiding operation failures by quickly choosing an alternate path.

While most scenarios include different paths to traverse from start to end, some processes have multiple endpoints. Some endpoints may represent error states, others may represent alternatives to the originally planned endpoint. In the latter case, it is often useful to include all acceptable endpoints into the analysis. Here, results may point out that it is beneficial to choose an endpoint that is more resilient or optimal in terms of operation based on the current scenario status. Graph algorithms can be executed with multiple destinations for this purpose. Algorithms specialized for multiple destinations may be available. Also, dynamic versions of SPF algorithms are an alternative (cf. section 2.3.1, p. 31f.). These algorithms directly integrate the graph updating procedure and may have advantages in the required calculation time. It is crucial to ensure that chosen algorithms work *fully dynamic*, allowing the addition and removal of vertices.

6.2.2 Non-Graph-based Decision-Making

Graph-based analyses show advantages in large, complex, or dynamic scenarios. They allow to extensively investigate process models at design time, optimizing their resilience before execution. At runtime, non-resilient process path segments may be identified early, resulting in process adaptations avoiding failing process operation. While

the graph-based approach is comprehensive and universal, it requires the translation of models to graphs.

Decision-making based on (weighted) decision matrices allows reducing the required effort. In the case of a resilience analysis, the path with the highest resilience value is chosen. If multiple criteria exist, the criteria and their values are added to the matrix. Weights allow to prioritize criteria against each other. Using a scalarization technique such as the WSM, scalar values are calculated for every alternative. The highest-ranked alternative is chosen. The procedure is similar to the scalarization of multi-criteria graphs illustrated in section 5.4.2 (p. 108ff.).

Applying weighted decision matrices for decision-making should be used with caution. The principle only includes parameters for the current decision, following process segments and their influences on process operation are not considered (i.e., it is a non-complete analysis). This can result in poor decisions, possibly leading to failing process operation. Figure 6.5 shows a process example heavily separated by gateways which makes it unsuited for weighted decision matrices. Achieving the best performance in such scenarios may require choosing not the best possible option at the current decision point to benefit from better options at later process segments. However, matrices seem to be a reasonable choice for processes avoiding the splitting of process paths. Processes in which the effects of decisions only apply to the current process segment are ideal. An example is illustrated by the process *Ex11* in Figure 6.6: decisions only last for the current process activity. Moreover, if a scenario includes a heavily dynamic behavior that leads to random changes of edge weights, a graph-based analysis will no longer have advantages compared to a matrices-based analysis.

Speeding up the identification and selection of alternatives is possible by using the neighbor table, summarizing connectivity information about opportunistic participants in the close proximity. Whenever a participant joins the neighborhood, a request for required functionality may be issued right away. Keeping the gathered information in an updated list and synchronizing it with the decision matrix, alternatives are directly available when they are needed. Delays due to gathering service information about available alternatives are avoided. If calling an alternative fails, it is removed from the list and another alternative is chosen. Figure 6.7 illustrates the speedup approach.

It is suggested to comprehensively investigate the usage of decision matrices at design time. The Max-Step analysis is well suited to compare graph-based results with matrices-based results. Operation of Max-Step is similar to decision matrices and can be easily compared to an all-paths or SPF analysis (cf. section 4.3.3, p. 87).

The usage of a graph-based and a WSM-based analysis approach is illustrated as part of the evaluation in sections 7.1.4 and 7.1.5 (p. 157ff.) as well as in the related proof-of-concept implementations in appendix A (p. 241ff.).

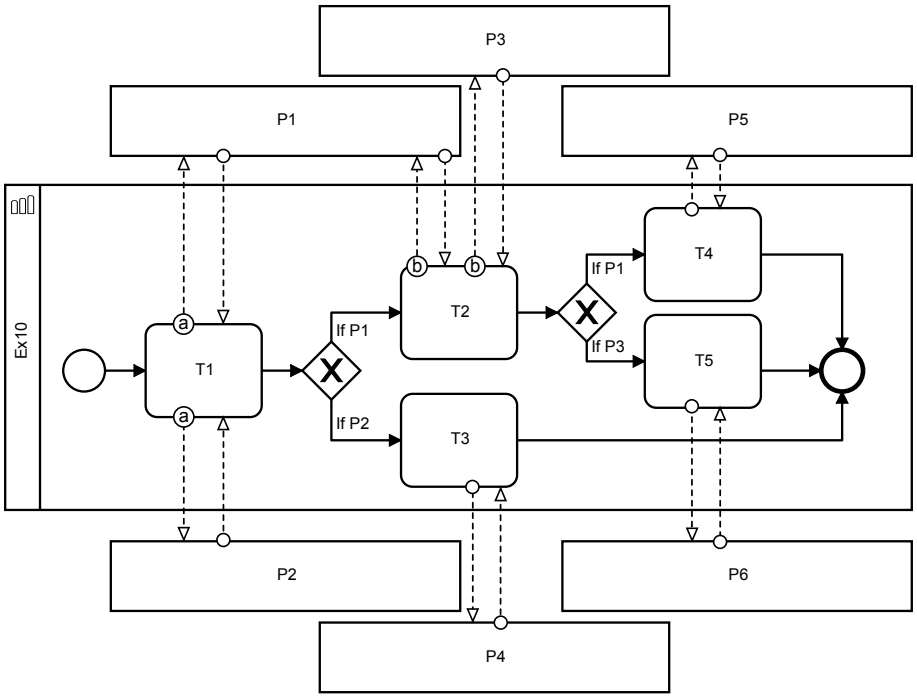

Figure 6.5: In process example *Ex10*, decisions have consequences for the remaining process path (*rBPMN*).

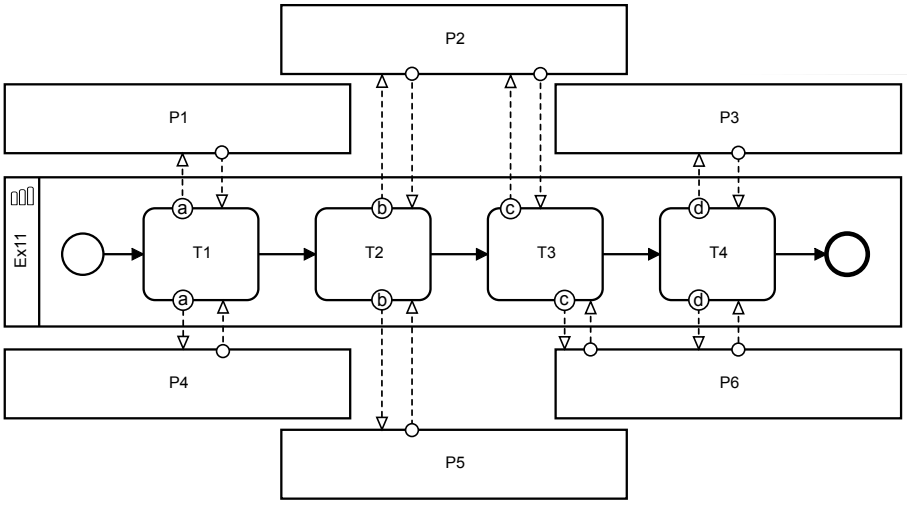

Figure 6.6: In process example *Ex11*, decisions are only of local relevance (*rBPMN*).

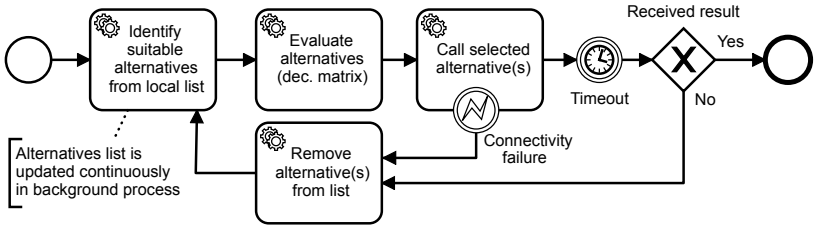

Figure 6.7: Speeding up identification and selection of alternatives (modeled in BPMN).

Summary

rBPMN includes new modeling elements to address dynamics in unreliable communication environments. Execution of processes requires technologies capable of implementing *rBPMN's* resilience strategies. Technologies based on the microservice- and container-paradigms are a sophisticated choice to realize the offering of movable functionality. The chapter elaborates approaches for the initial participant configuration and the identification and usage of functionality provided by neighboring participants.

Scenario dynamics demand the constant integration of updated edge weights as well as to add and remove vertices from the process graph. Recurring graph analyses trigger path adaptions avoiding non-resilient process segments. Decision-making based on weighted decision matrices is illustrated as an alternative for decisions based on graphs.

CHAPTER 7

EVALUATION AND RECOMMENDATIONS

Following the introduction of concepts and approaches for resilient models, graph-based analyses, and resilient execution in the previous chapters, this chapter evaluates the concepts and approaches using the agricultural scenario of an environmental-friendly slurry application. Based on the attempts of designing a resilient slurry process model using BPMN in chapter 2.2.4 (p. 25ff.), the process model is verified and optimized for resilient operation. Following, the slurry application process is extended to a multi-criteria optimization problem by adding the criteria accuracy, cost, and time along with resilience.

The execution of the slurry process is evaluated by two proof-of-concept implementations. The first implementation scenario is based on the mentioned multi-criteria optimization problem and uses graph-based decision-making to select a process path. The second implementation scenario adds a management participant to the slurry process and applies decision-making based on WSM. The software components are written in Java, being directly executable as self-contained microservices and as .jar-modules for the Camunda runtime engine [25]. All software used in the proof-of-concept implementations is open source and ready for adaptation and elaboration for other use cases and application domains.

The characteristics of the graph-based process analysis are investigated by measuring the performance and scalability of different graph algorithms. Since performance and scalability depend on the size of the tested business processes, a process graph

generator producing typical workflows of different sizes is created for the investigation. Further on, different metrics are evaluated for finding resilient process paths.

The chapter concludes by presenting recommendations for the modeling and execution of processes as well as for the resilience and multi-criteria analysis using *rBPMN's* concepts and approaches.

7.1 Evaluation of an Environmental-Friendly Slurry Application

The use case of an environmental-friendly slurry application is firstly introduced in section 2.1.4 (p. 14ff.), accompanied by agricultural background information. Process models implementing the slurry application are created as part of section 2.2.4 (p. 25ff.). Limitations of BPMN regarding the modeling of resilient processes and their optimal operation according to the scenario demands are illustrated (cf. Table 2.1, p. 30). In particular, this relates to all three approaches of modeling the sub-process *Analyze slurry* of Figure 2.3 (p. 26). With services for the creation of AppMaps, the slurry ingredients analysis, and the position sensing of the slurry spreader during the application, extensive communication with other participants is required.

This section extends the existing slurry process model up to generalized and executable process implementations, capable of being utilized for many different variants of slurry applications. The extension of the slurry process guides the evaluation of *rBPMN* regarding the objectives and challenges of this thesis (cf. section 1.2, p. 3ff.): Starting with the existing BPMN process model, section 7.1.1 verifies and optimizes the resilience using *rBPMN* elements. Following in section 7.1.2, additional criteria are integrated along with resilience, evaluating the multi-criteria analysis at design time. Section 7.1.3 elaborates the foundations of the two proof-of-concept implementations, which are evaluated in sections 7.1.4 and 7.1.5. Process model execution is realized by using the Camunda BPMN runtime engine in combination with the architectural principle of microservices, realizing the functionality offered by services of other participants. With process implementations using graph-based and WSM-based decision-making, the usage of *rBPMN* in combination with different decision-making techniques is evaluated. Figure 7.1 illustrates the evaluation sections, including the evaluation types and the addressed challenges.

The evaluation of the concepts and approaches for resilient models, graph-based analysis, and resilient execution is performed using theoretical and practical experiments. *Theoretical experiments* referrer to the design of process models and the manual analysis of resilient and multi-criteria operation using calculations. In contrast, the evaluation of *practical experiments* is performed by implementing and executing the

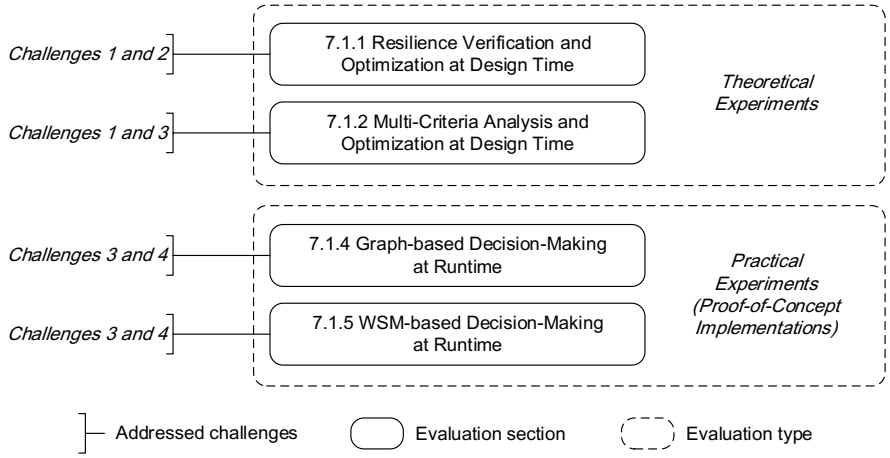

Figure 7.1: Evaluation sections with evaluation types and addressed challenges.

proof-of-concept. This allows the examination of process behavior during the actual execution of a process. A process may be investigated for its behavior while experiencing connectivity issues.

The evaluation objectives are stated at the beginning of every section. A summary of the evaluation results is provided in section 7.1.6, including a comparison of the differences between modeling resilient processes using BPMN and *rBPMN*.

7.1.1 Resilience Verification and Optimization at Design Time

This subsection investigates the following *evaluation objectives*:

Obj. 1: Evaluate the resilience verification of the existing BPMN slurry process model *S1* using *rBPMN-min* (discussed in chapters 3 and 4, addressing challenge 2).

Obj. 2: Evaluate the optimization of resilient operation using new modeling elements of *rBPMN* (discussed in chapter 3, addressing challenges 1 and 2).

In accordance with the evaluation objectives, the resilience verification of the existing slurry process model *S1* is performed subsequently. Connectivity characteristics of an exemplary scenario are used for the resilience analysis. Afterward, resilient operation is optimized by integrating new modeling elements, resulting in the *rBPMN* slurry process model *S2*. As evaluation criteria, *rBPMN* is assessed whether or not it is capable of verifying and establishing resilient process operation.

Resilience Verification of the Process Model

The model of process *S1* in Figure 2.3 (p. 26) consists of the three main parts *S1-a)* requesting an AppMap, *S1-b)* analyzing the slurry ingredients, and *S1-c)* applying the slurry to a field. All three parts communicate with other participants to realize their activities. Hence, process operation is vulnerable to the consequences of connectivity issues. Domain experts are not able to verify the resilience of the model since no verification method is provided by BPMN (cf. section 2.2.4, p. 25).

Existing BPMN process models starting at version 2.0 may be verified for resilience by integrating *rBPMN's OppMessageFlows* in combination with the corresponding QoS requirements and scenario-driven connectivity characteristics (⇒ *rBPMN-min*). After the integration of *OppMessageFlows* into process *S1*, the graph required for the resilience analysis is translated using the rule set elaborated in section 4.2 (p. 69ff.). This results in the resilience graph depicted in Figure 7.2. Since *S1* solely includes alternative process paths in *S1-b)*, the graph is represented by a chain of vertices with three separate paths for the graph analysis segment. The three process parts are combined by the two glue vertices *G1* and *G2*. While process model *S-DMN* depicted in Figure 2.6 (p. 28) is used for the process-to-graph translation of *S1*, the alternative models *S-ERR* (Figure 2.4, p. 27) and *S-GW* (Figure 2.5, p. 28) would result in a similar resilience graph. Here, the failure endpoints would have been removed from the models since communication failures will be identified by the resilience analysis.

Figure 7.3 presents the result of a resilience analysis on *S1*, following the LPF analysis approach elaborated in section 4.3.1 (p. 85f.). The graph edge weights R_e represent connectivity characteristic values for the corresponding message flows. For this scenario, applicable edge weights are defined as $R_e \in \mathbb{R} | 0 \leq R_e \leq 3$, resilience is

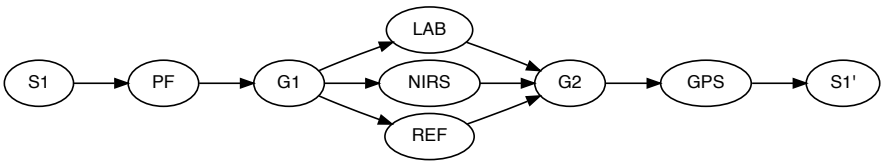

Figure 7.2: Resilience graph of *S1*.

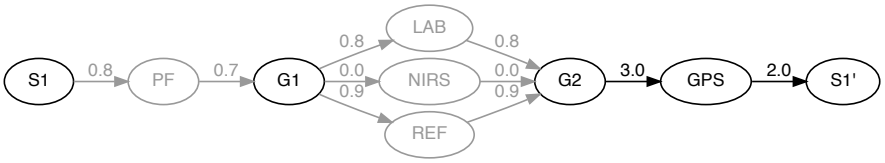

Figure 7.3: Resilience analysis on the graph of *S1*, with gray-colored non-resilient edges.

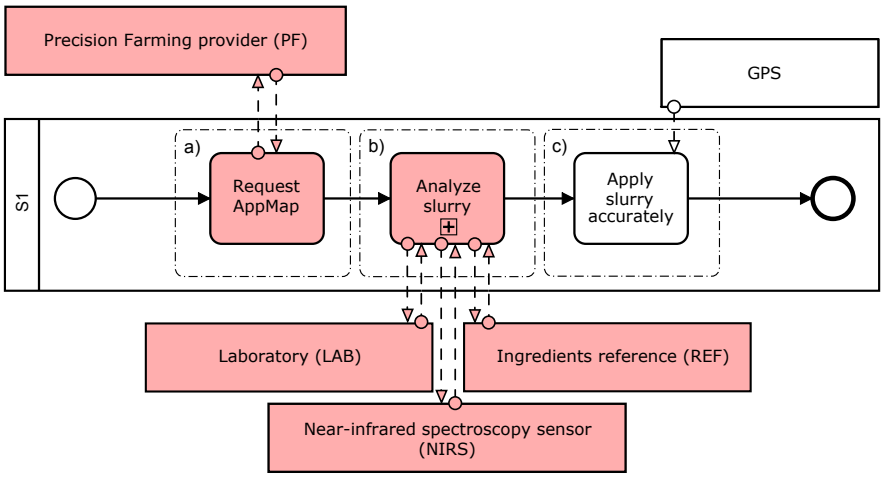

Figure 7.4: Result of the resilience analysis presented in Figure 7.3, visually integrated into the process model of *S1* (BPMN). Non-resilient process parts are colored in red.

given for $R_e \geq 1$. As illustrated by the gray-colored non-resilient edges in Figure 7.3, communication with all participants in *S1-a)* and *S1-b)* fails. Due to the use of the *GPS* technology, there is no need to request a position signal in *S1-c)* using a message flow (cf. Figure 2.3, p. 26). Hence, the incoming edge of *GPS* is always resilient and is set to the maximum weight value of 3. The reception of the *GPS* signal is identified as resilient with a value of 2. However, no resilient process paths could be found using a Dijkstra algorithm on the graph of *S1*. The chosen scenario would end in a process breakdown at runtime. Figure 7.4 illustrates the analysis visually integrated into the process model of *S1*, allowing domain experts the straightforward visual identification of failing process parts. Failing activities and message flows are colored in red.

Resilience Optimization of the Process Model

A resilient process model of *S1* can be achieved by using *rBPMN's* concept of movable functionality. AppMaps with basic accuracy may be created automatically without manual optimization of human experts. Moving functionality for the creation of AppMaps from the precision farming provider to the slurry spreader allows continuous operation in case of connectivity issues. Likewise, a basic version of an ingredients reference service may allow the provisioning of adequate ingredients analysis results even in the case of no connectivity. The process model *S2* depicted in Figure 7.5 presents an optimized version of *S1*, using new modeling elements introduced by *rBPMN* (cf. section 3.3, p. 47ff.). Movable functionality has been included in process parts *S2-a)* and *S2-b)* by using *MovParticipants* and *OppTasks*. Seamlessly and opportunistically

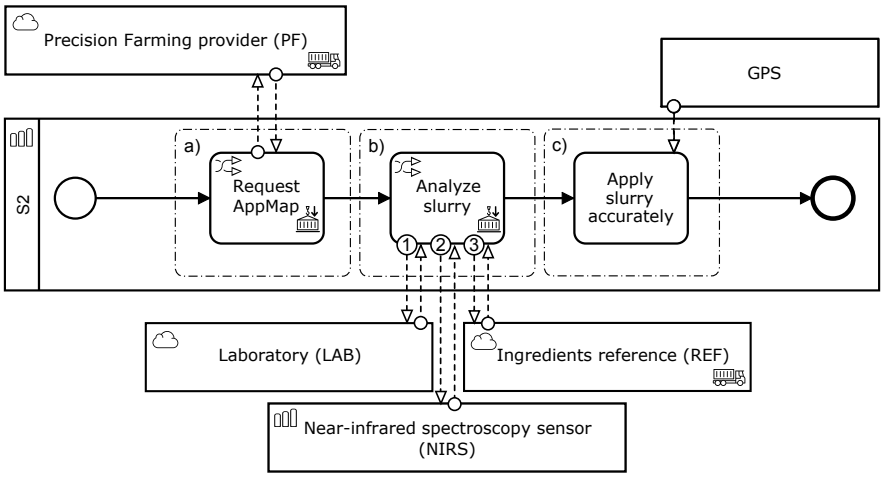

Figure 7.5: Slurry process *S2* (*rBPMN*).

connected participants are easily identifiable by the added connectivity attributes in the upper left corner of a participant (cf. Figure 7.5).

A drawback of the process model *S1* is that the three alternatives for the slurry analysis are only identifiable when taking a closer look at the corresponding sub-process *Analyze slurry*. By inspecting the process model of *S1* with the collapsed sub-process, only message flows to *LAB, NIRS,* and *REF* are identifiable, but not that they are alternatives to each other. The model may be misunderstood by assuming that every participant has to be available for resilient operation. In process model *S2* depicted in Figure 7.5, this has been eliminated by replacing the *OppMessageFlows* with *OppPriorityFlows*. A static ordering of alternatives is configured, prioritizing *LAB* following *NIRS*, *REF*, and the locally moved version of *REF*, labeled as *REF(L)*. This is realized by the *OppPriorityFlows* with the priorities *1, 2, 3* and the *OppTask* of the *Analyze slurry* activity, defined in the process model (cf. Figure 7.5). The process graph of *S2* is illustrated in Figure 7.6. It basically represents the graph of *S1*, updated with separated paths for the locally moved functionality in process parts *S2-a)* and *S2-b)*.

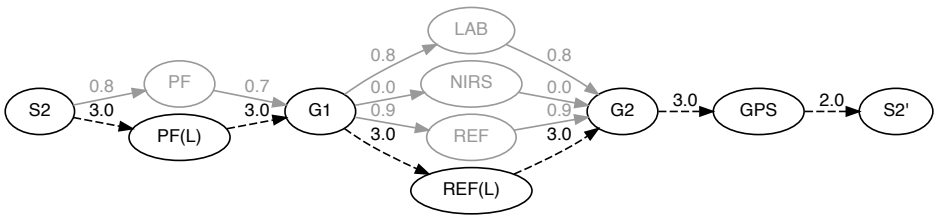

Figure 7.6: Resilience analysis on the graph of *S2*, with gray-colored non-resilient edges.

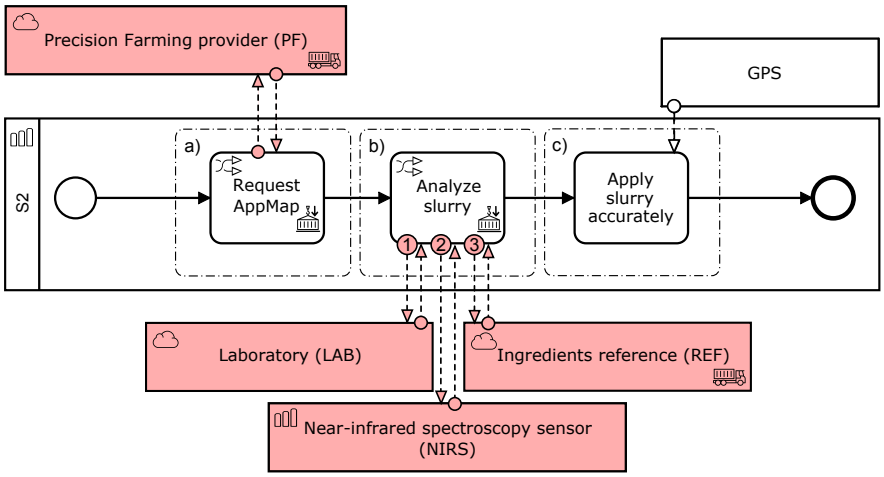

Figure 7.7: Visually integrated result of the resilience analysis on slurry process *S2* (*rBPMN*). Non-resilient process parts are colored in red.

An analysis of *S2* confirms the resilience of the process model. Dashed edges in Figure 7.6 depict one remaining path, including the locally moved functionality for the AppMap creation and the slurry analysis. In Figure 7.7, the process parts *S2-a)* and *S2-b)* are no longer colored in red, indicating resilient operation.

The *evaluation results* of this subsection are summarized as follows:

- Slurry process model *S1* is identified as non-resilient, using *rBPMN-min* and a resilience analysis (⇒ fulfilling *Obj. 1*).
- The newly introduced slurry process model *S2* operates resiliently using movable functionality of *rBPMN* (⇒ fulfilling *Obj. 2*).
- Usage of *OppPriorityFlows* enhances the semantics of process model *S2*.

7.1.2 Multi-Criteria Analysis and Optimization at Design Time

This subsection investigates the following *evaluation objectives*:

Obj. 3: Illustrate *rBPMN's* capability to optimize a slurry process model for its use in different slurry application variants (discussed in chapter 3, addressing challenge 1).

Obj. 4: Evaluate the capabilities of *rBPMN* to design an optimal operating slurry process regarding a set of different criteria (discussed in chapters 3 and 5, addressing challenges 1 and 3).

Obj. 5: Evaluate the multi-criteria operation of the slurry process at design time (discussed in chapter 5, addressing challenges 3).

Following, the process model *S2* is generalized regarding the demands of different slurry applications. Specifying criteria priorities and metric thresholds for a newly modeled process is often challenging. Hence, a comparison-based multi-criteria analysis is performed subsequently, identifying process behavior in a typical scenario at design time. The analysis results allow optimizing the criteria set for process execution, ensuring to comply with the demands of domain experts in different application scenarios. *rBPMN* is assessed regarding its capability to provide the required modeling elements, analysis approaches, and multi-criteria metrics to fulfill the evaluation objectives.

A Generalized Process Model for Slurry Applications

The process model *S2* is identified as resilient for the given scenario and its connectivity characteristics. A single resilient process path is found. Other slurry application scenarios may face superior connectivity conditions, resulting in multiple resilient paths to be chosen from. Here, the process needs to be configured to select the best-suited process path according to a defined criteria set.

Besides, the process model *S2* has to be generalized to be applicable for many different variations of slurry application scenarios. For example, slurry scenarios exist that avoid the partfield-driven application of slurry and use fixed slurry application amounts for an entire field. If using AppMaps, an integration of different precision farming providers for the creation of AppMaps is desired, serving the needs of different farmers. The slurry analysis in *S2-b)* should allow to dynamically integrate other types of ingredients analyses, e.g., rapid testing methods. The accuracy of position sensing using GPS is questionable in combination with partfield-driven AppMaps. Safety margins to creeks and streams may be reduced with more accurate position sensing technology without the danger of violating legal requirements.

Figure 7.8 presents a generalized slurry process *S3* using modeling elements of *rBPMN* (cf. section 3.3, p. 47ff.). A new exclusive gateway in part *S3-a)* determines whether or not precision farming is used. If so, the process continues with parts *S3-b)* and *S3-c)*. If not, no communication with other participants is required to finish the slurry application. The process is directed to an activity that manually applies the slurry to the field in *S3-c)*. Hence, this path is always resilient in terms of communication.

The *OppMessageFlow* requesting an AppMap at the precision farming provider has been replaced with an *OppDecisionFlow*, accompanied by an *OppDynTask*. This allows the integration of other precision farming providers and to dynamically choose the appropriate alternative based on defined criteria. The *OppTask* for the slurry analysis in *S2-b)* has been replaced with an *OppDynTask*, allowing the integration of other participants dynamically at runtime. The *OppDecisionFlows* calling *LAB,*

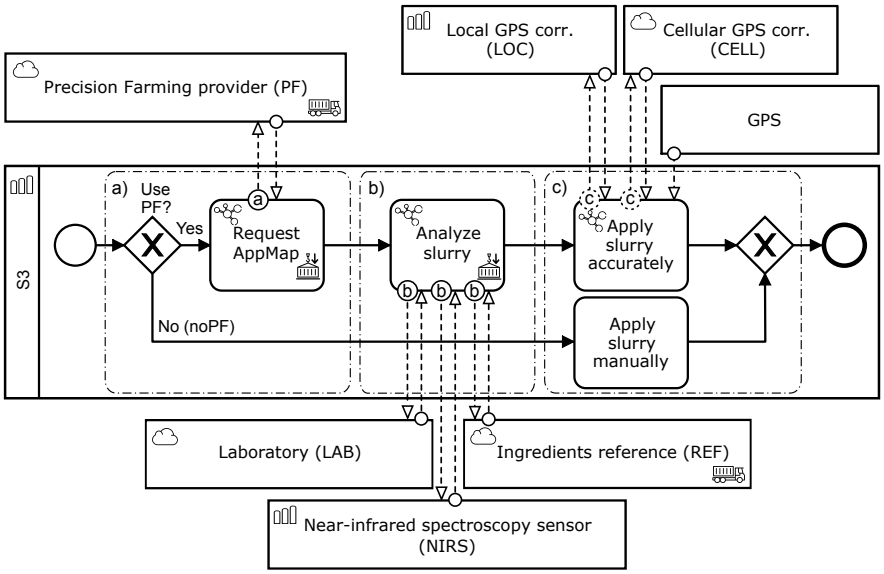

Figure 7.8: Slurry process *S3*, representing a multi-criteria optimization scenario (*rBPMN*).

NIRS, and *REF* ensure selection of the best-suited alternative based on a multi-criteria analysis. In *S3-c)*, two participants for the correction of the *GPS* signal have been added. Based on the RTK technology, the correction signal may be transferred from a remote station using cellular communication or by a local station in the proximity of the field. Both *OppDecisionFlows* are defined as optional since not every scenario requires an increased position sensing accuracy. Integration of dynamic participants for the position correction is ensured by the *OppDynTask*, realizing the *Apply slurry accurately* activity (cf. Figure 7.8).

Multiple criteria are required for selecting an optimal process path in this evaluation. The criterion accuracy describes how accurately an activity realizes its assigned task. For a slurry application using process model *S3*, it may be used as an indicator for the environmental-friendliness of process operation. By dividing a field into partfields representing growth potentials based on soil and yield information, over-fertilization and groundwater contamination can be avoided (cf. [86]). Besides the effects on the environment, typical criteria for slurry applications are economical aspects such as cost and time of process operation.

Following, the newly introduced process model *S3* is analyzed for its characteristics in an exemplary scenario.

Comparison-based Multi-Criteria Analysis of the Process Model

The criteria set chosen for the subsequent multi-criteria analysis is illustrated in Table 7.1. Resilience, accuracy, and cost are defined as 1. level criteria, requiring to meet the defined criteria metrics (cf. section 5.2, p. 97f.). Time is defined as a 2. level criteria, resulting in selecting the path with minimum time effort from the remaining paths fulfilling the 1. level criteria. However, the current criteria definition in Table 7.1 is incomplete: The threshold values to be met by the 1. level criteria are missing. Since their definition may be challenging for newly modeled processes, a multi-criteria analysis is performed prior to their further definition.

The process graph for the multi-criteria analysis is depicted in Figure 7.9. It has been created by using the multi-criteria process-to-graph translation rules introduced in section 5.3 (p. 98ff.). Compared to the graph of *S2*, a path avoiding precision farming is added. Furthermore, part *S3-c)* requires to include position correction using a cellular-based service (*CELL*) and a local-based service (*LOC*) as optional *OppDecisionFlows* into the graph. Since *GPS* is mandatory for position sensing, it is part of every path. After the *GPS* vertex, the process may end directly or apply position correction. The separate paths for *CELL* and *LOC* are integrated using the glue vertex *G3*. *G3* facilitates the application of an outgoing edge weight for *GPS* as well as incoming edge weights for *CELL* and *LOC*.

While Figure 7.9 represents the correct translation of *S3*, the graph structure may be simplified. In Figure 7.10, three separate paths for the position sensing in *S3-c)* exist. The mandatory *GPS* is combined with the paths of *CELL* and *LOC* to common vertices, simplifying the graph structure. It is worth mentioning that for these vertices also the edge weights of *GPS* and *CELL/LOC* have to be combined to common edges. From here, *CELL* and *LOC* refer to the combinations of *GPS+CELL* and *GPS+LOC*.

Criteria graphs including example edge weights for the criteria resilience, accuracy, cost, and time are shown in Figures 7.11 to 7.14. The resilience edge weights in Figure 7.11 represent connectivity characteristics for requesting and receiving information from participants (incoming and outgoing edges of a participant-representing vertex). A

Table 7.1: Criteria set and importance levels for the design-time analysis of *S3*.

Importance level	**1. Level**	**2. Level**	**3. Level**
Level type	Optimization (1^{st} tier)	Optimization (2^{nd} tier)	Monitoring
Semantic: Meeting criteria demands is ...	required	desired	negligible
Criteria	Resilience Accuracy Cost	Time (min)	-

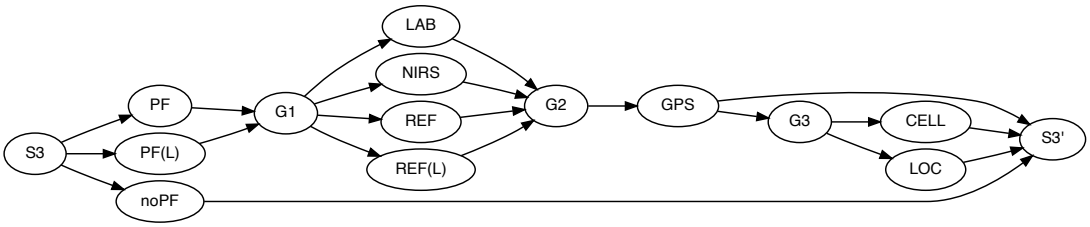

Figure 7.9: Process graph of *S3*.

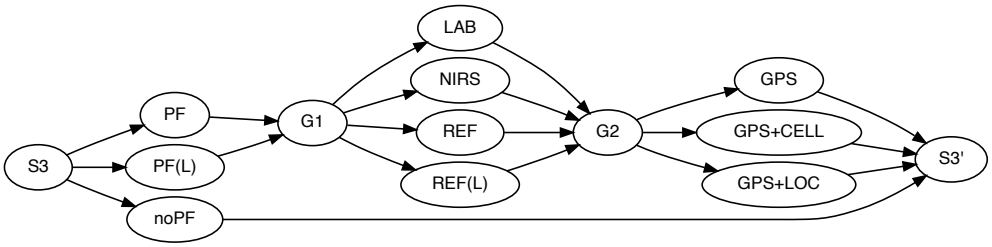

Figure 7.10: Simplified process graph of *S3*.

resilient edge is defined as $C_e^R \geq 1$. Accordingly, all edges are resilient in the given graph. Edge weights for accuracy, cost, and time are bound to an activity. Values are defined as $C_e^x \in \mathbb{R} | 0 \leq C_e^x \leq 1$.

The accuracy of non-precision-farming processes is lower compared to the use of an automatically or manually created AppMap. However, in regards to costs and time, a manually created AppMap is more expensive and time-consuming than an automatically created or an unused AppMap. In *S3-b)*, more accurate slurry analysis mechanisms are more cost-intensive than less accurate mechanisms. In regards to operational time, automated procedures are faster than manual mechanisms such as a laboratory analysis. More accurate position sensing adds cost to the process operation. While local position correction stations ensure high accuracy and communication resilience, they require a little more time effort for setting up. In typical slurry process scenarios, the demanded path for the given scenario is defined as a resilient path with high accuracy and low cost and time efforts.

A comparison-based analysis provides extensive information about reasonable process paths and metrics (cf. section 5.5.2, p. 113). Since *S3* represents a newly modeled process and its behavior is unknown, a comparison-based analysis is performed at design time. This allows to choose criteria metrics and to specify thresholds for the 1. level criteria accuracy and cost, supporting the automation of the analysis for process execution.

Table 7.2 lists all possible process paths with selected metrics. The Pareto-optimal paths include the alternative for not using precision farming. This path is always

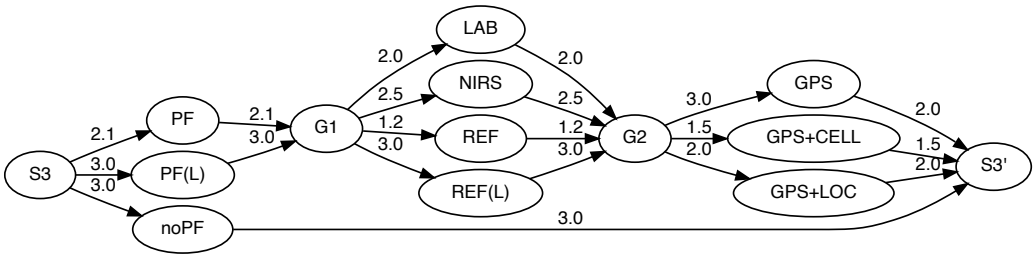

Figure 7.11: Resilience graph of *S3*.

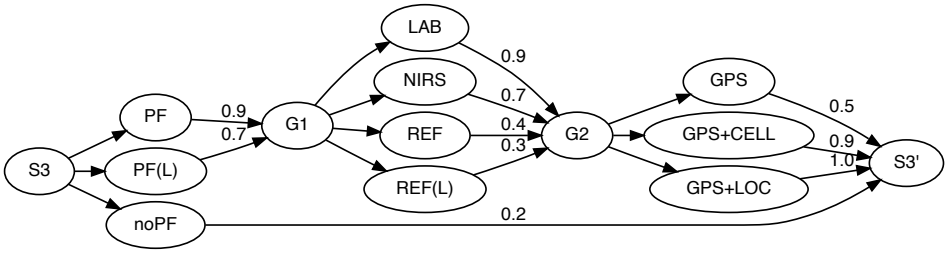

Figure 7.12: Accuracy graph of *S3*.

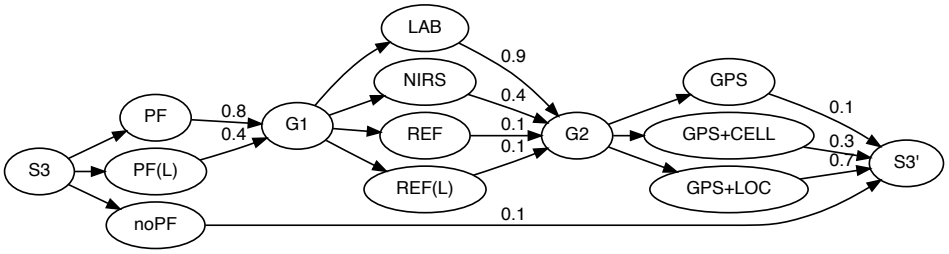

Figure 7.13: Cost graph of *S3*.

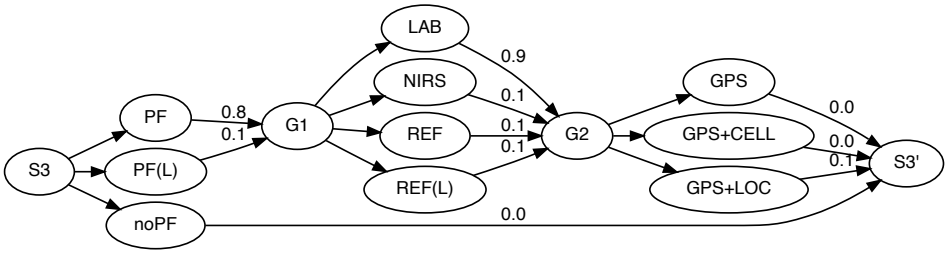

Figure 7.14: Time graph of *S3*.

Table 7.2: Full path list of *S3* in a comparison-based multi-criteria analysis.

#	**Path variation**	**Resilience**	**Accuracy**	**Cost**	**Time**	
	S3 → ... → *S3'*	C_l^R	C_l^A	C_t^C	C_t^T	**Pareto?**
1.	*PF* → *LAB* → *GPS*	2.0	0.5	1.8	1.7	-
2.	*PF* → *LAB* → *CELL*	1.5	0.9	2.0	1.7	✓
3.	*PF* → *LAB* → *LOC*	2.0	0.9	2.4	1.8	✓
4.	*PF* → *NIRS* → *GPS*	2.0	0.5	1.3	0.9	-
5.	*PF* → *NIRS* → *CELL*	1.5	0.7	1.5	0.9	-
6.	*PF* → *NIRS* → *LOC*	2.0	0.7	1.9	1.0	-
7.	*PF* → *REF* → *GPS*	1.2	0.4	1.0	0.9	-
8.	*PF* → *REF* → *CELL*	1.2	0.4	1.2	0.9	-
9.	*PF* → *REF* → *LOC*	1.2	0.4	1.6	1.0	-
10.	*PF* → *REF(L)* → *GPS*	2.0	0.3	1.0	0.9	-
11.	*PF* → *EF(L)* → *CELL*	1.5	0.3	1.2	0.9	-
12.	*PF* → *REF(L)* → *LOC*	2.0	0.3	1.6	1.0	-
13.	*PF(L)* → *LAB* → *GPS*	2.0	0.5	1.4	1.0	-
14.	*PF(L)* → *LAB* → *CELL*	1.5	0.7	1.6	1.0	-
15.	*PF(L)* → *LAB* → *LOC*	2.0	0.7	2.0	1.1	-
16.	*PF(L)* → *NIRS* → *GPS*	2.0	0.5	0.9	0.2	✓
17.	*PF(L)* → *NIRS* → *CELL*	1.5	0.7	1.1	0.2	✓
18.	*PF(L)* → *NIRS* → *LOC*	2.0	0.7	1.5	0.3	✓
19.	*PF(L)* → *REF* → *GPS*	1.2	0.4	0.6	0.2	✓
20.	*PF(L)* → *REF* → *CELL*	1.2	0.4	0.8	0.2	-
21.	*PF(L)* → *REF* → *LOC*	1.2	0.4	1.2	0.3	-
22.	*PF(L)* → *REF(L)* → *GPS*	2.0	0.3	0.6	0.2	✓
23.	*PF(L)* → *REF(L)* → *CELL*	1.5	0.3	0.8	0.2	-
24.	*PF(L)* → *REF(L)* → *LOC*	2.0	0.3	1.2	0.3	-
25.	*noPF*	3.0	0.2	0.1	0.0	✓

Declaration: ✓⇒ Pareto-optimal path - ⇒ no Pareto-optimal path

Table 7.3: Selected paths of *S3* used in a comparison-based analysis of criteria graphs.

#	**Path variation**	**Resilience**	**Accuracy**	**Cost**	**Time**	
	S3 → ... → *S3'*	C_l^R	C_l^A	C_t^C	C_t^T	**Pareto?**
2.	*PF* → *LAB* → *CELL*	1.5	0.9	2.0	1.7	✓
3.	*PF* → *LAB* → *LOC*	2.0	0.9	2.4	1.8	✓
5.	*PF* → *NIRS* → *CELL*	1.5	0.7	1.5	0.9	-
6.	*PF* → *NIRS* → *LOC*	2.0	0.7	1.9	1.0	-
14.	*PF(L)* → *LAB* → *CELL*	1.5	0.7	1.6	1.0	-
15.	*PF(L)* → *LAB* → *LOC*	2.0	0.7	2.0	1.1	-
17.	*PF(L)* → *NIRS* → *CELL*	1.5	0.7	1.1	0.2	✓
18.	*PF(L)* → *NIRS* → *LOC*	2.0	0.7	1.5	0.3	✓

Declaration: ✓⇒ Pareto-optimal path - ⇒ no Pareto-optimal path

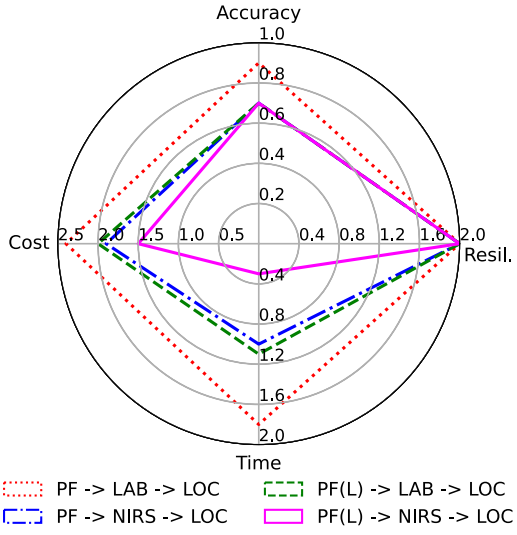

Figure 7.15: Radar chart with selected *LOC*-paths of *S3*.

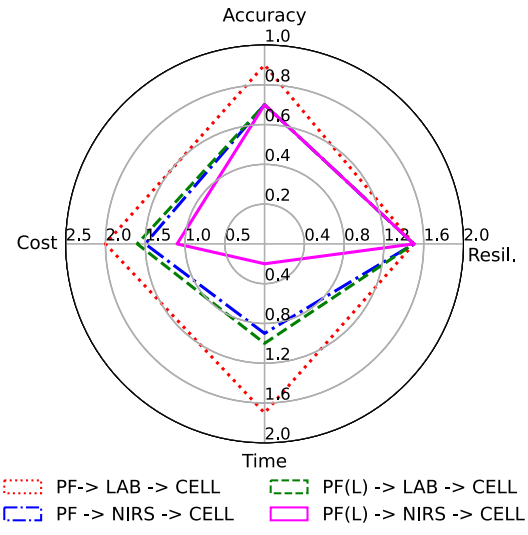

Figure 7.16: Radar chart with selected *CELL*-paths of *S3*.

resilient and has the lowest cost and time requirements. However, the minimum accuracy of the path is low with a value of $C_l^A = 0.2$. The Pareto-optimal paths number 2 and 3 include the highest minimum accuracy $C_l^A = 0.9$ of all paths.

The resilience level C_l^R indicates that all paths are resilient. However, due to possible connectivity deviations in real-world process executions, the minimum resilience level is configured as $C_l^R \geq 1.5$, filtering paths including *REF* from the results list. Since the objective is an environmental-friendly slurry application, the minimum average accuracy of operation is defined as $C_l^A \geq 0.7$. Table 7.3 lists the remaining paths, qualified for the chosen minimum metrics for resilience and accuracy.

A visual representation of the remaining paths may help filtering more paths. An illustration of the remaining paths using radar charts is depicted in Figures 7.15 and 7.16. The figures indicate the correlation between cost and time. A less costly path also reduces the required operation time. Visual illustrations may help domain experts to determine whether or not the current model is sufficiently optimized for the envisaged scenarios and application domains.

Optimization of the Criteria Set for Process Execution

Subsequently, aspects of the design-time multi-criteria analysis are recapitulated to optimize the criteria set for process execution. The minimum metrics defined for resilience and accuracy represent static thresholds: they do not adapt to the characteristics of different scenarios. This way, it is ensured that the metric requirements are met in ever specific scenario the process may be used for. If no alternatives are available fulfilling the metrics, the process will be shown as failing at design time. For real-world process executions, failing is not the desired reaction. Instead, a process supervisor should be informed that the process is not able to fulfill the required metrics. The available options should be illustrated, supporting the supervisor with his decision. For instance, a supervisor may continue a process with some limitations instead of risking a process breakdown. However, it is important to include the supervisors' decision since the process is not able to solve the issues without violating the defined metrics.

Edge weights of $C_e^R \geq 1.5$ have been set as a threshold for resilient communication in this multi-criteria analysis. The inclusion of an extra margin for resilience (here 0.5) is helpful to eliminate issues at runtime based on false or inaccurate connectivity estimations. An analysis at design time will identify process segments prone to connectivity failures due to poor resilience. Domain experts may strengthen the segments before executing the process. While extra resilience margins are a tool for process modeling, most scenarios will remove the margins during execution. Extra margins potentially exclude participants that have been identified as resilient during runtime. In the worst

Table 7.4: Importance levels for the criteria set of *S3*, to be used for process execution.

Importance level	**1. Level**	**2. Level**	**3. Level**
Level type	Optimization (1^{st} tier)	Optimization (2^{nd} tier)	Monitoring
Semantic: Meeting criteria demands is ...	required	desired	negligible
Criteria	Resilience ($C_l^R \geq 1.0$) Accuracy ($C_l^A \geq 0.7$) Cost ($C_t^C \leq 2.0$)	Accuracy (max, 50 percent) Cost (min, 30 percent) Time (min, 20 percent)	-

case, a multi-criteria analysis may face no available alternatives in its graph although resilient communication with participants is possible.

Since participants providing the required functionality are added dynamically to the sets of available alternatives at runtime, more cost-intensive or time-consuming options for the creation of AppMaps, the slurry ingredients analysis, and the position sensing may exist than originally planned. As long as there are less expensive and time-consuming alternatives available, additional cost and time will be avoided. However, if costly and time-consuming dynamic alternatives are the only available options, the process would choose them and continue.

Process owners are able to control this behavior without losing the option to include participants dynamically at runtime. First, a required threshold is set by defining a criterion as of 1. level importance in conjunction with a criteria metric. Afterward, the same criterion is added as of 2. level importance. For instance, a maximum path cost of $C_t^C = 2.0$ is set for the 1. level importance. Subsequently, the same criterion is set as of 2. level importance with instructions for its optimization. Table 7.4 illustrates an updated version of the importance level definition to be used for process execution. For the resilience, accuracy, and cost criteria, threshold weights are defined as of 1. level importance. In addition, the criteria accuracy, cost, and time have been defined as of 2. level importance with 50 percent for accuracy over 30 percent for cost and 20 percent for time. This more precise definition helps to better control the best-suited path selection, especially in the case of an automated analysis.

The *evaluation results* of this subsection are summarized as follows:

- Process *S3* represents a flexible *rBPMN* process model, to be used in different slurry applications using criteria set definitions for optimal operation (⇒ fulfilling *Obj. 3* and *4*).
- A comparison-based multi-criteria analysis has analyzed a typical slurry scenario, supporting the definition of the criteria set for process execution (⇒ fulfilling *Obj. 5*).

7.1.3 Process Execution using BPMN and Microservices

Aspects regarding the execution of *rBPMN* process models are discussed subsequently. Furthermore, technical background information regarding the two proof-of-concept implementations is provided.

Executing *rBPMN*-based Process Models

Several BPM software manufacturers offer their products and services on the market. Many allow the execution of BPMN process models by interpreting the sequence and message flows and by linking the modeled activities to related source code, realizing the process' objectives. Most manufacturers create product suites, where a BPMN runtime engine is combined with additional tools for modeling, monitoring, execution, and support. Some engine manufacturers distribute (parts) of their products as open source, others require users to purchase licenses for their products.

The execution of *rBPMN* process models requires the runtime engine to understand *rBPMN's* extension elements. The creation or extension of an engine may require considerable software development effort, not reasonable if only a subset of the elements is used or the number of concerned processes is small. An alternative is the use of an existing BPMN runtime engine and to implement *rBPMN's* resilience strategies for modeling and execution as part of the source code of activities. However, this limits process modeling elements to BPMN elements supported by the chosen runtime engine. As an advantage, only the required resilience strategies relevant for the chosen processes have to be implemented for process execution.

The approach of using an existing BPMN runtime engine has several drawbacks. Resilience verification at design time is no longer available, preventing improvements of the model. Definitions of alternative groups and priorities chosen by domain experts are lost. Alternatives and locally moved functionality are no longer visually identifiable. This illustrates that an *rBPMN*-capable runtime engine remains the primary solution for comprehensive process executions.

The two proof-of-concept evaluations of *rBPMN's* concepts and approaches apply the Community Edition of the Camunda Platform to bypass the implementation of an *rBPMN*-capable runtime engine. The code required for the realization of identification, selection, and usage of dynamic alternatives is provided by reusable classes. The classes are enclosed in the Java .jar-file as part of the Camunda process modules.

The architectural principle of microservices is an ideal fit for the realization of process participants. Self-contained modules comply with the need for moving functionality, communication using interfaces avoids the restriction of implementation technologies along different participants. The Spring-Framework [138] supports the creation of

microservices and is used in the implementations. By utilizing Camunda and Spring, two of the most popular technologies for BPM and microservices are applied to illustrate *rBPMN's* resilience strategies in practice.

Technical Information regarding the Proof-of-Concept Implementations

The proof-of-concept comprises two different slurry processes, both taking place in unreliable communication environments. The following sections 7.1.4 and 7.1.5 elaborate the process models and their participants in detail. Both processes represent multi-criteria optimization scenarios, including the criteria communication resilience, accuracy, cost, and time of participants and their activities. Decision-making for alternatives is either based on a multi-criteria graph analysis or on WSM, depending on the evaluation scenario (cf. section 6.2, p. 131ff.).

The proof-of-concept illustrates the realization of process participants as BPMN runtime engine modules, as standalone microservices, and as a combination of both, a runtime engine encapsulated in a self-contained microservice (cf. section 6.1.2, p. 126f.). Participants provide RESTful services to the slurry spreader process for the realization of its activities. The services disseminate their functionalities using service metadata. Different configuration profiles allow using the same microservices for the execution in cloud environments and as locally moved functionality.

Spring Eureka allows clients to register their services in a service registry, ready to be used by service-seeking participants. However, Eureka is designed for cloud environments not comparable to unreliable communication environments. The proof-of-concept implementations provide a custom service discovery mechanism, allowing to find services across unreliably connected Eureka servers (cf. section 6.1.4, p. 129f.).

The reasonable effort for setting up an unreliable communication environment with different participants, their configuration and services is inconvenient for interested parties to examine the proof-of-concept. Hence, a neighbor-service is included, managing available neighbors in a controllable neighbor table (cf. section 6.1.3, p. 127ff.). This allows surveying the proof-of-concept on a single computer system. Besides, adapting the neighbor-service to interface proactive MANET routing algorithms such as OLSR (cf. 2.1.3, p. 12f.) enables to use the proof-of-concept in an unreliable execution environment.

A set of REST-calls is provided in a collection, importable into the program *Postman* [140]. Using Postman, the execution of the proof-of-concept can be controlled by restarting processes, by adding/deleting neighboring participants, and by inspecting currently available services. Also, examples for the initial set-up sequence configuration file and the design of service functionality interfaces with JSON-based data transfer objects are provided as part of the proof-of-concept implementations.

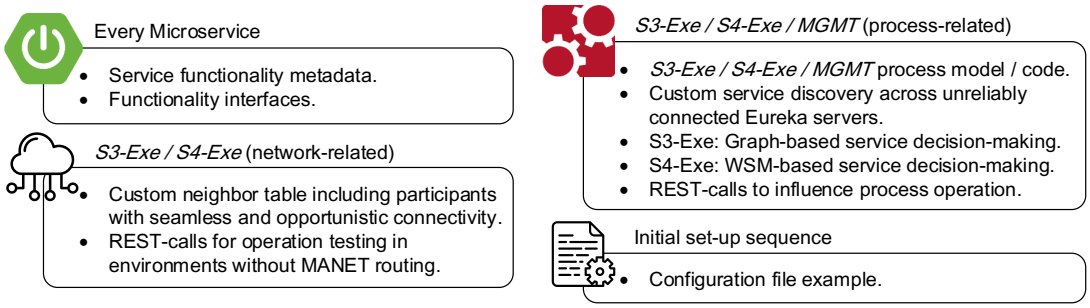

Figure 7.17: Software pieces implemented for the evaluation scenarios *S3-Exe* and *S4-Exe*.

Fig. 7.17 summarizes the software pieces that have been implemented as part of the two proof-of-concept implementations, serving as templates for other application domains and use cases. The source code is available on Github [119]. Interested parties may execute the implementations illustrated subsequently by following the instructions provided in appendix A (p. 241ff.).

7.1.4 Graph-based Decision-Making at Runtime

This subsection investigates the following *evaluation objectives*:

- *Obj. 6:* Illustrate the customizability of the multi-criteria analysis for a real-world scenario (discussed in chapter 5, addressing challenge 3).
- *Obj. 7:* Evaluate the graph-based multi-criteria analysis at runtime (discussed in chapter 5, addressing challenges 3 and 4).
- *Obj. 8:* Evaluate the use of BPMN and microservices for the implementation of *rBPMN* process models (discussed in chapters 6 and 7, addressing challenge 4).
- *Obj. 9:* Evaluate *rBPMN's* resilience strategies for the movement of functionality, the discovery of neighbors, and the on-demand usage of functionality in unreliable environments (discussed in chapter 6, addressing challenge 4).

A scenario-based multi-criteria analysis approach is required to solve the scenario requirements of this subsection. The approach is elaborated subsequently. Further on, the architecture and graph-based implementation of the proof-of-concept is outlined, followed by an illustration of the service discovery and decision-making at process runtime.

Scenario-based Multi-Criteria Analysis at Runtime

The first proof-of-concept evaluation scenario is represented by the multi-criteria optimization problem elaborated in section 7.1.2. With the importance level defined in Table 7.4, hard thresholds for the criteria resilience, accuracy, and cost have been defined as of 1. level importance. Furthermore, accuracy and cost are also defined as of 2. level importance along with the time criterion and a weighting indication (prioritization) for accuracy (50 percent), cost (30 percent), and time (20 percent).

The design-time analysis in section 7.1.2 applied a comparison-based analysis approach by using an all-paths search and selected metrics to identify considerable paths. The comprehensive metric set of the importance-level-definition (cf. Table 7.4) is challenging for an automated runtime analysis procedure. An iteration-based approach would include the removal of unqualified graph edges for the criteria resilience, accuracy and cost (cf. section 5.5.1, p. 112f.). Afterward, an SPF search on the joint graph of the remaining criteria weights for accuracy, cost, and time may be applied (section 5.4.2, p. 108ff.). This is straightforward for the resilience criterion, where the service discovery identifies available participants at runtime. Unqualified edges of the accuracy criterion are removed by simply verifying each edge of the accuracy graph for its accuracy value. However, the challenge is to verify whether or not the total path cost remains lower or equal to the defined maximum value of $C_t^C \leq 2.0$. Since this metric is calculated by summarizing path weights, the result is a path and not a specific edge. Hence, an edge removal as required by the iteration-based approach is not possible.

A solution would be to check the total path cost after searching for the shortest path in a joint graph of accuracy, cost, and time. The creation of a joint graph is possible by inverting accuracy weights in the manner of $1 - C_x^A$. However, if the path has a higher total cost as desired, no decision can be made. An alternative would be to use a comparison-based analysis and filter all unqualified paths. Afterward, a decision for one of the remaining paths may be performed by joining the remaining edges and applying an SPF analysis. This illustrates the need for a scenario-based analysis approach for this scenario.

The approach applied in the proof-of-concept creates a joint graph by integrating criteria values for accuracy, cost, and time (section 5.4.2, p. 108ff.). Following, the graph is analyzed by using a *KSP algorithm* to identify a ranked list of shortest paths (cf. section 2.3.1, p. 31ff.). The highest-ranked path is checked for the thresholds of minimum accuracy and the total path cost. If both thresholds are met, the path is chosen. If not, the metric verification continues with the second highest-ranked path, until a satisfying path is found. As an alternative, unqualified accuracy edges can be removed from the graph prior to the KSP search. The chosen path will be the same whether minimum accuracy is checked before or after the KSP search.

Scenario Architecture and Implementation

Figure 7.18 provides an architectural overview of the scenario and its collaborating participants. Several service-offering participants are placed within the cloud, where connectivity is supposed to be reliable. A precision farming participant *PF* offers a service to calculate AppMaps, *LAB* and *REF* provide slurry ingredients analyses and *CELL* allows to correct a GPS position for the application of slurry. A cellular gateway connects the cloud with the slurry spreader and its process *S3-Exe*, located on an agricultural field. Part of the slurry spreader are the two locally moved services of *PF(L)* for the calculation of AppMaps and *REF(L)* for a local slurry analysis. *NIRS* and *LOC* are located in the proximity of the slurry spreader. Communication on the field is realized by a MANET, connectivity issues in communicating with *NIRS* and *LOC* may occur. The service-offering participants are implemented as microservices using Spring Boot, avoiding the use of BPMN process models and runtime engines. The slurry process *S3-Exe* is executed using the Camunda BPMN runtime engine, encapsulated in a self-contained microservice on the slurry spreader.

Execution of the multi-criteria slurry process *S3* defined in section 7.1.2 using the Camunda runtime engine is facilitated by removing all *rBPMN* elements from the process model, resulting in the process *S3-Exe* depicted in Figure 7.19. Thereon, the existence of locally moved services is no longer visible in the model. This may lead to misunderstandings of domain experts and process workers not familiar with the process model implementation. For ease of implementation, the decision of whether or not precision farming is used is combined with the decision for an AppMap created manually by a precision farming expert *PF* or automatically by the locally moved

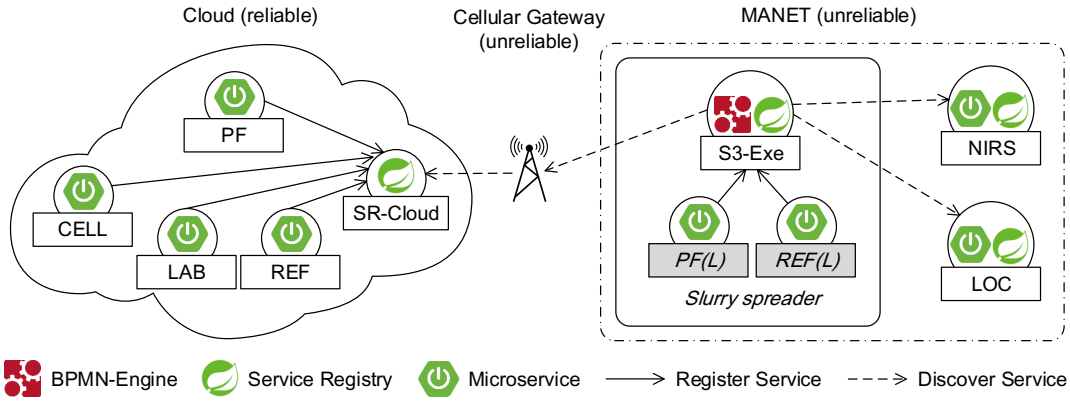

Figure 7.18: Architectural overview of the process *S3-Exe* and its participants. Services are registered, discovered, and used in an unreliable communication environment.

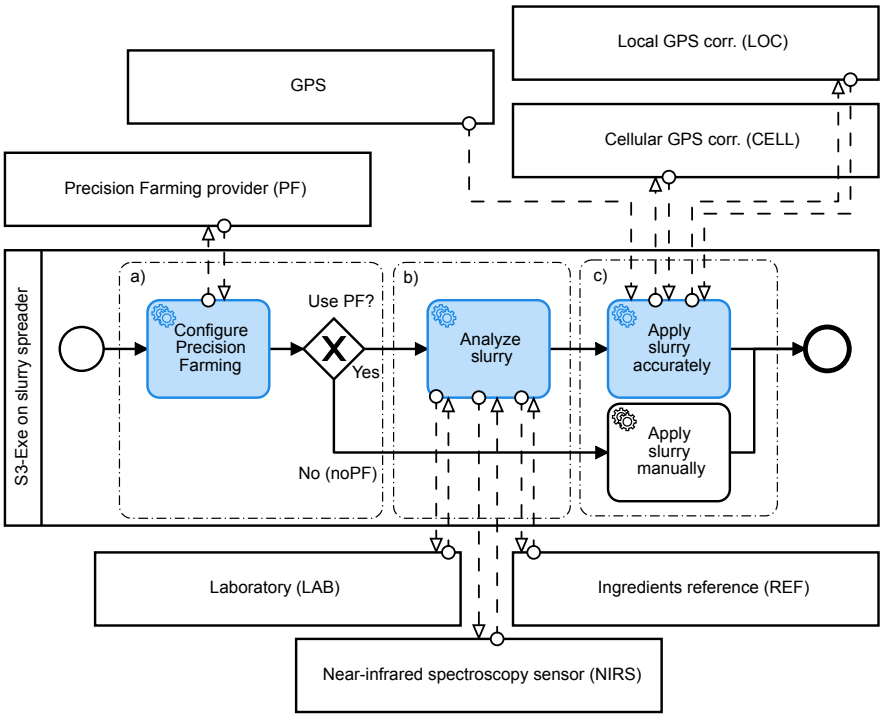

Figure 7.19: Executable process *S3-Exe*, deciding on alternatives for the blue-colored activities using graph-based decision-making (modeled in BPMN).

PF(L) service with less precision. Keeping the process structure of *S3* would have resulted in two decision-making steps instead of one.

The major contributions of this proof-of-concept scenario are the implementations for decision-making, consisting of a service discovery applicable for unreliable communication environments and the graph-based analysis for the best-suited process path. The package diagram of *S3-Exe* in Figure 7.20 shows the three sub-packages *Services*, *Delegates*, and *Functions*. Service discovery and graph-based decision-making are implemented in classes of the *Functions* sub-package, being used by the delegate classes of the blue-colored activities of Figure 7.19. Besides, the *RestStart* class of the *Service* sub-package allows starting the process using a RESTful interface while *SpApplication* defines the package as a Camunda application.

Service Discovery and Decision-Making at Runtime

Service discovery is essential for decision-making since it allows to dynamically update the process graph with available services and their criteria metadata. Figure 7.18 illustrates the existence of four *Service Registry servers (SR)* in the scenario, realized by using Spring Eureka. According to the approach elaborated in section 6.1.4 (p. 129f.),

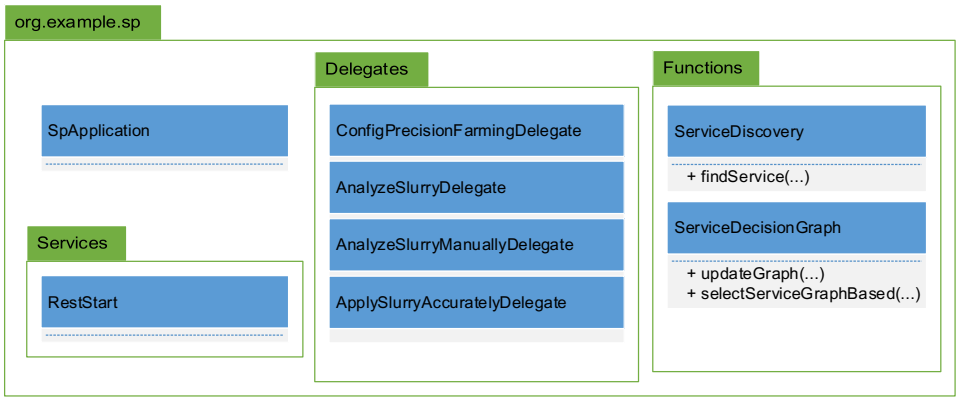

Figure 7.20: Package diagram of *S3-Exe*, including sub-packages and classes.

every participant with opportunistic connectivity maintains its own SR. Another SR is placed in the cloud, managing service information of the seamlessly connected participants. Every time there is a need to discover services, the *ServiceDiscovery*-class of the *Functions* sub-package requests the currently available neighbors from the neighbor-service and queries each of them for the requested service type. Every SR provides a list of registered services and the corresponding metadata. With Eureka, service metadata may be defined as part of the application configuration. Listing 7.1 depicts an excerpt of the configuration of the *PF* service. The configuration profile defines its name, the server port and path for accessing the service as well as the service type and the criteria values for accuracy, cost, and time. Furthermore, the configuration states the seamlessly connected cloud as the service location.

Listing 7.1: Profile setting of the Spring Boot application *PF*.

```
1   spring:
2     profiles: pf
3   server:
4     port: 8024
5   eureka:
6     instance:
7       metadata-map:
8         type: "PF"
9         location: "cloud"
10        accuracy: "0.9"
11        cost: "0.8"
12        time: "0.8"
13        id: "PF"
14        urlanalysis: "/analysis"
```

The Eureka replica mechanism facilitates a high availability of Eureka servers by replicating the registries of different servers with each other. While the replica mechanism is designed for cloud environments, it may be an alternative to query every individual SR in the scenario. Since all services of the neighbor registries are present in the local service registry, a single query to the local registry is sufficient to find all service alternatives. Since participants may be seen only for short periods in unreliable environments, the neighbor table may be used to dynamically adapt the Eureka replica configuration for appearing and disappearing participants. Figure 7.21 illustrates the procedure of dynamically adapting the replica configuration with appearing and disappearing neighbor participants.

The query response of available services represents the list of alternatives to be decided on. Using the services' metadata, the process graph (the same as depicted in Figure 7.10) is updated with criteria information (weights) of every service. Unavailable services are removed from the graph. The process graph represents a joint graph of the criteria accuracy, cost, and time. After updating the graph, KSPs are identified, according to the scenario-based multi-criteria analysis approach described in this subsection. The highest-ranked path fulfilling the criteria thresholds for minimum path accuracy and total path cost is selected. Afterward, the selected service is called using the addressing information provided in its metadata.

Listing 7.2 depicts an excerpt of the decision-making during the *Configure Precision Farming* activity of *S3-Exe*: During decision-making, a KSP analysis is performed, identifying 25 available paths. The highest-ranked path #0 fails to meet the required minimum path accuracy and is dropped. Then, the second highest-ranked path #1 is verified for minimum path accuracy and total path cost with positive results and is selected.

The *evaluation results* of this subsection are summarized as follows:

- A scenario-based multi-criteria analysis approach using a KSP algorithm fulfills the scenario requirements for optimal operation (⇒ fulfilling *Obj. 6* and *7*).
- Execution of the slurry process *S3-Exe* proofed *rBPMN's* capability to cope with the challenges of unreliable communication by dynamically adapting operation to the given circumstances (⇒ fulfilling *Obj. 7* and *8*).
- During process execution of *S3-Exe*, *rBPMN's* resilience strategies enabled the dynamic discovery of neighboring participants as well as the movement and on-demand usage of functionality (⇒ fulfilling *Obj. 9*).

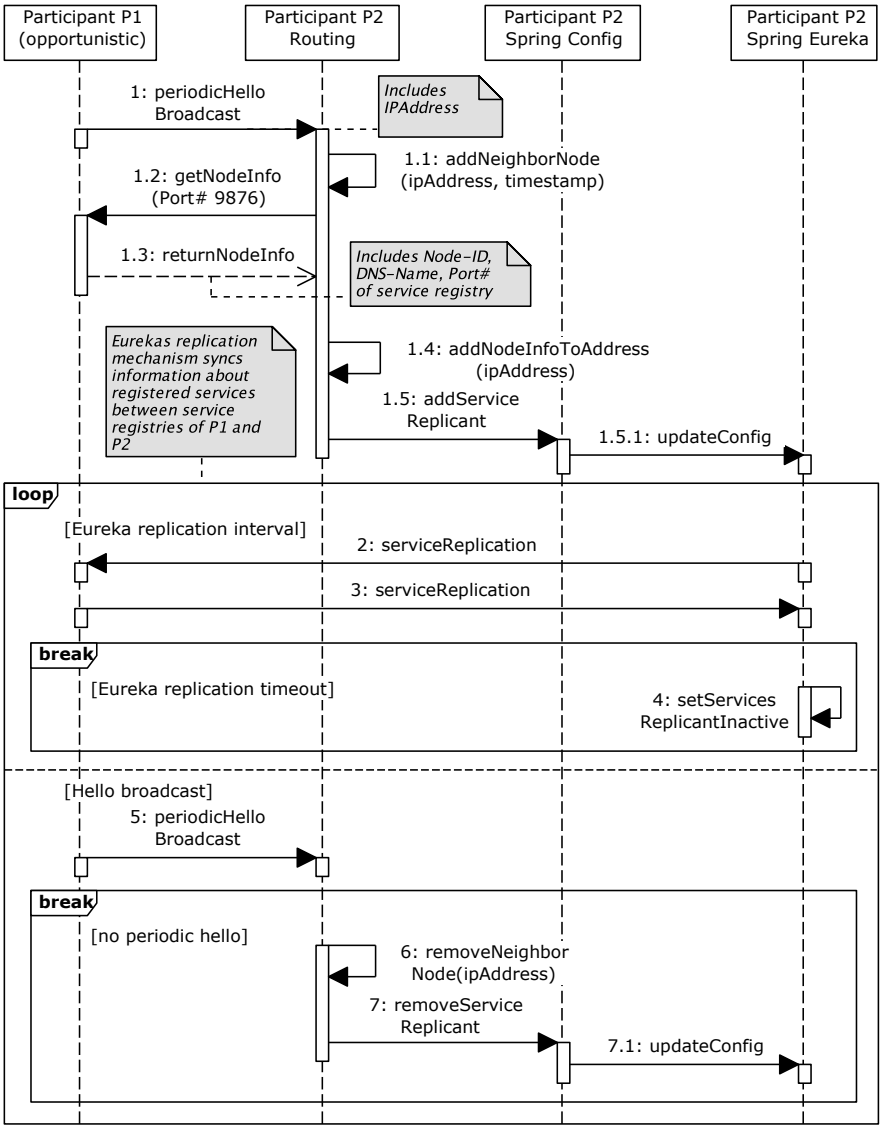

Figure 7.21: The Eureka replica mechanism allows to query a single service registry for all available services of the slurry spreader and its collaborative participants.

Listing 7.2: Graph-based selection of a path at the *Configure Precision Farming* activity of process *S3-Exe*.

```
--- k-shortest-paths analysis ---
Number of shortest paths (S to S') found: 25
       yenKShortestPath-#0: [S, noPF, S']
       yenKShortestPath-#0-Weight: 0.43
       yenKShortestPath-#1: [S, PF(L), G1, NIRS, G2, CELL, S']
       yenKShortestPath-#1-Weight: 0.74
       yenKShortestPath-#2: [S, PF(L), G1, REF, G2, CELL, S']
       yenKShortestPath-#2-Weight: 0.8
       Omitting remaining paths...
Checking minimum accuracy (min defined: 0.3) and
total cost (max defined: 2.0) of path #0 ...
       Minimum accuracy: 0.2
       Total cost: 0.1
Checking minimum accuracy (min defined: 0.3) and
total cost (max defined: 2.0) of path #1 ...
       Minimum accuracy: 0.7
       Total cost: 1.1
Path selected: #1
       Path: [S, PF(L), G1, NIRS, G2, CELL, S']
       Minimum accuracy 0.7 >= 0.3 (dMinAccuracy)
       Path cost 1.1 <= 2.0 (dCostLimit)
```

7.1.5 WSM-based Decision-Making at Runtime

This subsection investigates the following *evaluation objectives*:

Obj. 10: Evaluate the use of a WSM-based decision-making approach in combination with an *rBPMN* process model (discussed in chapters 3 and 6, addressing challenge 4).

Obj. 11: Evaluate the integration of dynamically appearing participants as additional alternatives originally not part of the process model (discussed in chapters 3 and 6, addressing challenges 3 and 4).

A slurry application variant including a *management participant (MGMT)* is the basis of this evaluation section. At first, the slurry scenario of the second proof-of-concept implementation is outlined, along with the scenarios' architecture and implementation. Following, decision-making based on WSM is illustrated.

Scenario Architecture and Implementation

The second proof-of-concept implementation scenario includes a slurry spreader that is managed by an *MGMT* participant located in the cloud. As illustrated in Figure 7.22, *MGMT* assigns the slurry spreader a slurry task, monitors and controls the application, and eventually documents the process after finishing the application. At the process' start, the slurry spreader receives the task and drives to the field. During the application of slurry, status data is sent to the *MGMT* participant, which may adjust process operation. Process *S4-Exe* on the slurry spreader dynamically decides on the services used for the slurry analysis and the position sensing. Due to the avoidance of *rBPMN* modeling elements, the locally moved services of *MGMT(L)* and *REF(L)* are not visible in the process model of Figure 7.22.

The architectural overview depicted in Fig. 7.23 illustrates the existence of two dynamically appearing participants not part of the process model. In addition to the alternatives *NIRS*, *REF* and *REF(L)*, another service *REF2* may be used for the slurry ingredients analysis. Services for correcting the *GPS* signal are provided by *CELL*, *LOC*, and an additional *CELL2* participant. As in the previous scenario, the

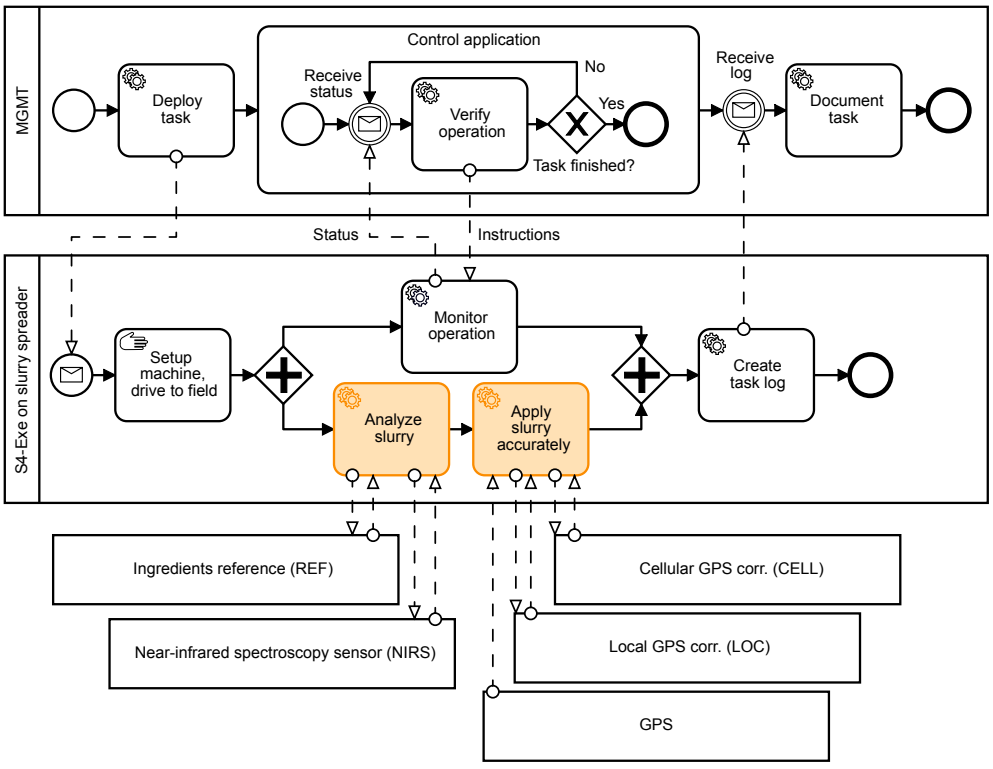

Figure 7.22: Executable process *S4-Exe*, deciding on alternatives for the yellow-colored activities using WSM-based decision-making (modeled in BPMN).

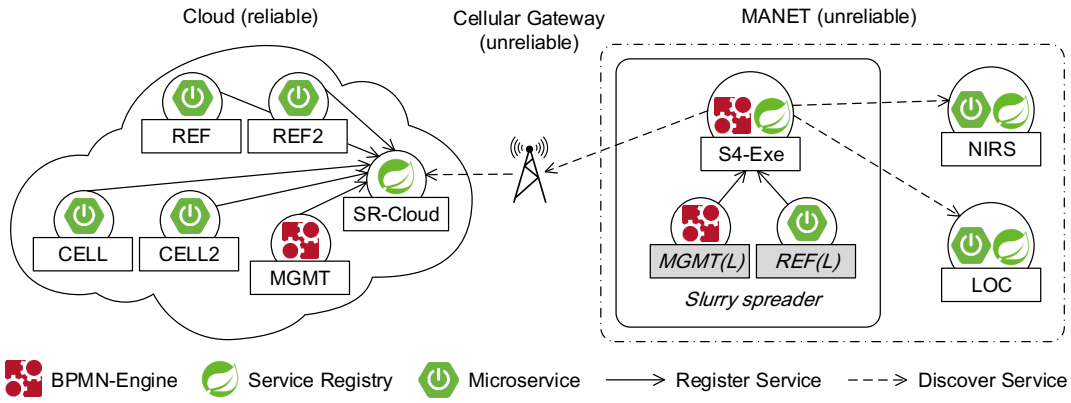

Figure 7.23: Architectural overview of the process *S4-Exe*. Participants *REF2* and *CELL2* appear dynamically at process runtime.

services register at four different SR. The process of *MGMT* is executed using the Camunda runtime engine. The sub-process *Control application* has been duplicated into a runtime environment, encapsulated in a microservice. This results in *MGMT(L)* as movable functionality to control the slurry application on the slurry spreader in case of connectivity issues.

Service Discovery and Decision-Making at Runtime

Service discovery across unreliable connected participants is realized by the same *ServiceDiscovery*-class as in the previous scenario. Since queries are issued for finding a specific type of functionality, every available service offering the functionality is identified. Hence, the dynamically appearing participants *REF2* and *CELL2* also serve as an alternative since they are part of the query response.

The activity *Monitor operation* of the process model *S4-Exe* depicted in Figure 7.22 decides whether the central *MGMT* participant placed in the cloud or the locally moved service *MGMT(L)* manages the slurry application. Because *MGMT(L)* is considered as a backup, *MGMT* is preferred if available. The yellow-colored activities of *S4-Exe* decide on the services for the slurry analysis and the position sensing/correction. Here, a WSM-based analysis is applied, realized by the class *ServiceDecisionWSM*. Therefore, the metadata information for the criteria accuracy, cost, and time of every entry of the available services list is extracted. Following, values for accuracy are inverted to be harmonized with cost and time. Afterward, the values are weighted and summarized to a resulting value. The service with the lowest summarized value is selected and called by the process *S4-Exe*.

Listing 7.3: WSM-based selection of a slurry analysis service in process *S4-Exe*.

```
1        Weight | Service0 Service1 Service2... (last line:
2        summarization of criteria values for a service )
3        0.6 0.7 0.6 0.8 0.3
4        0.3 0.0 0.1 0.0 0.4
5        0.1 0.1 0.1 0.0 0.1
6        ***
7        0.0 0.4 0.4 0.5 0.3
```

Listing 7.3 depicts an excerpt of the proof-of-concepts' decision-making on the service for the slurry analysis: The first column specifies the weights of the WSM, followed by four identified services (here: *REF2*, *REF*, *REF(L)*, and *NIRS*) in the subsequent columns. Every line represents a different criteria, following the order of accuracy, cost, and time. The final line lists the weighted and summarized results for every service.

The *evaluation results* of this subsection are summarized as follows:

- Execution of the slurry process *S4-Exe* proofed *rBPMN's* capability to integrate an alternate decision-making based on WSM (⇒ fulfilling *Obj. 10*).
- During process execution of *S4-Exe*, *rBPMN's* resilience strategies enabled the discovery and usage of dynamically appearing participants originally not part of the process model (⇒ fulfilling *Obj. 11*).

7.1.6 Findings and Remarks

During this section, the initial slurry application process has been extended up to the two process implementations *S3-Exe* and *S4-Exe*, representing the proof-of-concept. Considering actual slurry applications performed in agriculture, processes may be larger, including additional participants. For instance, before *MGMT* deploys a task to a slurry spreader in *S4-Exe*, a farmer may instruct an agricultural contractor to perform the slurry application using its machinery and workers. As outlined in [120], trucks may support the application by transporting the slurry to the corresponding fields. Finally, the applied slurry amounts have to be reported to a local authority. However, these steps may be modeled in additional processes, complementing the scenario illustrated by *S4-Exe*.

Furthermore, there is a need for ad-hoc slurry applications without extensive planning in BPM tools. In these scenarios, the slurry application is started by docking a tractor to a slurry spreader, picking up the slurry, and driving to the field. Using information of the tractors' machine bus, the composition of tractor and slurry spreader may be analyzed for its technical capabilities (e.g., the capability of applying slurry to

partfields, cf. [120]). This may be used to initiate and configure a slurry process model, controlling and documenting the application process. *S3-Exe* may be employed as such a process model, covering many different variants of slurry applications.

While there may be minor differences depending on the executing parties, the slurry application process models represent realistic scenarios. Hence, the process implementations *S3-Exe* and *S4-Exe* of the proof-of-concept are considered as real-world slurry applications. This also applies to the chosen criteria set, including accuracy, cost, and time of process operation.

Both of the practical slurry evaluation scenarios show resilient operation, regardless of the unreliable communication environment faced on the agricultural field. The slurry spreader is capable of adjusting and continuing its operation during runtime. Participants and their services are dynamically discovered and integrated into the decision-making, leading to an optimal process operation within the realms of possibilities.

The following paragraphs summarize the evaluations findings in the environmental-friendly slurry application scenarios regarding *rBPMN's* concepts and approaches. Aspects of process modeling, decision-making, and process execution are discussed.

Process Modeling

The attempt to model a resilient slurry process using BPMN in section 2.2.4 (p. 25ff.) illustrates several drawbacks due to the exposure of the process to an unreliable communication environment. *rBPMN* allows verifying the existing slurry model regarding resilient operation by using a subset of *rBPMN*, referred to as *rBPMN-min* (cf. section 3.4, p. 57).

Adapting the slurry process using *rBPMN's* modeling elements for the integration of alternatives and the movement of functionality across participants allows to optimize and guarantee resilient operation. Besides communication resilience, the process may operate optimally with regards to other process criteria such as accuracy, cost, and time.

Implementing slurry process *S3* in BPMN emphasizes the importance of visually meaningful process models. *S3-Exe* misses illustrating the existence of locally moved functionality. Groups of alternatives for the decision-making on services for the slurry analysis and the position correction are not visible. Users get a false impression about the process' workflow.

Table 7.5 extends Table 2.1 (p. 30) by adding the subset *rBPMN-min*, the full set of extension concepts *rBPMN-max* (cf. section 3.4, p. 57) and the evaluation results to the comparison of process modeling aspects. As illustrated, *rBPMN-min* is able to improve the comprehensibility of process models regarding unreliable communication. Additionally, it enables the verification of process models for resilient

Table 7.5: Comparison of modeling resilient processes using BPMN and *rBPMN*.

Aspect	*S-Err*	*S-GW*	*S-DMN*	*rBPMN-min*	*rBPMN-max*
Use of existing BPMN / DMN elements.	+	+	+	-	-
Graphical expressiveness of unreliable connectivity, alternatives and priorities.	o	o	-	+	+
Avoidance of additional graphical hierarchies for the modeling of unreliable communication characteristics.	o	o	o	o	+
Ability to visually indicate possibly failing message flows and to integrate the corresponding QoS requirements / connectivity-characteristics.	-	-	-	+	+
Ability to adapt operation for unplanned runtime dynamics (e.g., fluctuating connectivity, appearing / disappearing participants).	-	-	-	-	+
Ability of the model to adapt for the operation demands and connectivity characteristics of different scenarios.	-	o	o	-	+
Ability to continue process operation in case of no connectivity.	-	-	-	-	+
Simplicity of modeling resilient processes.	-	-	-	+	+
Ability to verify process resilience before runtime.	-	-	-	+	+
Summarized points	4	5	4	8	16

Declaration: + ⇒ full support / 2 points,
o ⇒ limited support / 1 point, - ⇒ no support / 0 points

operation. Especially concerning the integration of process dynamics for optimal operation, *rBPMN-max* should be used. *rBPMN-max* is capable of integrating dynamically appearing participants and adjusting process operation optimally to the given scenario conditions.

Graph-based and WSM-based Decision-Making

The graph-based approaches to verify resilient operation at design-time and to decide for the best-suited process path at runtime work well in the evaluation scenario. The multi-criteria optimization problem of *S3* shows limitations when considering the iteration-based analysis approach since the metric of the total path cost C_t^C is unable to eliminate any edges from the cost graph. The scenario-based analysis approach applied in this section illustrates its versatility by adapting the analysis to the characteristics of the slurry scenario.

The proof-of-concept evaluations demonstrate the suitability of decision-making based on graphs as well as based on WSM for the slurry scenarios. Both approaches

require some effort to initially develop the source code, which may be reused afterward. While WSM works well in the process *S4-Exe*, this is not necessarily the case in other scenarios. As illustrated in section 6.2.2 (p. 133), WSM does not perform a complete analysis. Hence, it is not able to consider follow-up decisions in the process path.

Process Execution

Ensuring resilient process operation by extending the source code linked to activities is effective for using a given BPMN runtime engine such as Camunda. Integration of supporting frameworks and technologies such as Spring is advised. Spring Boot eased the creation of microservices for service-offering participants while Spring Eureka helps to implement service discovery across different participants in an unreliable communication environment. The resilience strategies for the discovery of neighboring participants using a neighbor table and the on-demand usage of functionality show effectiveness (cf. sections 6.1.3 and 6.1.4, p. 127ff.).

Service discovery using the Eureka replica mechanism was evaluated with ambivalent results. While the mechanism reduces the querying effort, additional communication overhead is created. This may not be necessary due to a missing need for service discovery.

Besides, the continuous appearance and disappearance of services from participants quickly triggers the circuit-breaker pattern [71] of Eureka. The circuit-breaker removes services from the registry which have numerous timeouts in a specified period. During practical testing of the proof-of-concept implementations, this has led to the unavailability of services in the SR. As a result, the processes have been unable to continue operation and failed. While the circuit-breaker pattern is useful for reliable cloud environments, it is misleading in many unreliable communication environments with dynamic participants. As a solution, such scenarios may continue to query every available SR of neighboring participants at the time of a service request.

7.2 Evaluation of the Graph-based Process Analysis

While a graph-based analysis of a process is able to consider characteristics of entire paths from the process' start to end, it requires some computation time on the analyzing device. This section evaluates the performance and scalability characteristics of graph analyses. For this purpose, experiments using different graph algorithms are performed on a performance-limited device. Further on, the applicability of different graph algorithms and metrics for the resilience analysis is evaluated. The section begins by illustrating the evaluation setup and the generation of process graphs used during evaluation.

7.2.1 Evaluation Setup

Business processes often include a variety of participants, geographically distributed across the area of application. The technical configuration of the participants / their devices may be represented by cloud systems, PCs, smartphones, sensors, and actuators. Since many processes in unreliable communication environments use performance-restricted devices, the evaluation is performed on a *Raspberry Pi Zero WH* [144]. This Raspberry configuration features an ARM-1GHz-processor (BCM 2835 SOC, single core) and 512 MB of RAM.

The software measuring the performance requirements is written in Java and executed on the Raspberry using the OpenJDK Runtime Environment 1.8. The JGraphT-Library [95] provides the implementations for graph-based search algorithms used in the evaluation.

7.2.2 Generation of Process Graphs

Depending on the objectives and application areas, business processes differ in their characteristics in terms of structure and size.

While the concrete structure of different processes varies, most include multiple decision points. Based on the configuration and the value of process variables/parameters, one path or another is chosen. In a graph, these decision points are represented by multiple edges originating from a common vertex and splitting the path to the process' end. When analyzing graph-based search algorithms, the number of graph decision points is the major aspect. Path sequences which are avoiding decision points do not affect their performance significantly. Hence, these sequences are not considered in this evaluation.

A process graph generator has been designed and implemented for the performance evaluation of graph-based analyses. The generator builds up a graph by splitting paths

at each following vertex. To reflect the varying sizes of possible process graphs, *horizontal and vertical graph layers (H-Layers/V-Layers)* are introduced. H-Layers define how often a path is split (horizontally) at a vertex. In other words, H-Layers describe the number of outgoing edges of a vertex. Additionally, V-Layers configure the number of follow-up vertices in which the process path is split by outgoing edges. After the defined number of V-Layers, the following vertical layers are used to merge the previously separated paths again. This results in a DAG with a starting and an ending vertex, including a high number of different process paths.

During the graph generation, a random weight value $C_e^x \in \mathbb{R} | 0 \leq C_e^x \leq 1$ is assigned to every generated edge. The weights represent normalized values of a random criterion. If required, inversion of values allows to comply with SPF algorithms.

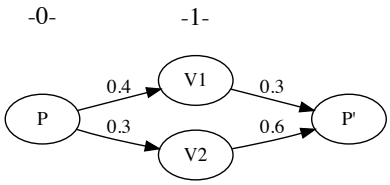

Figure 7.24: A generated process graph with 1 V-Layer.

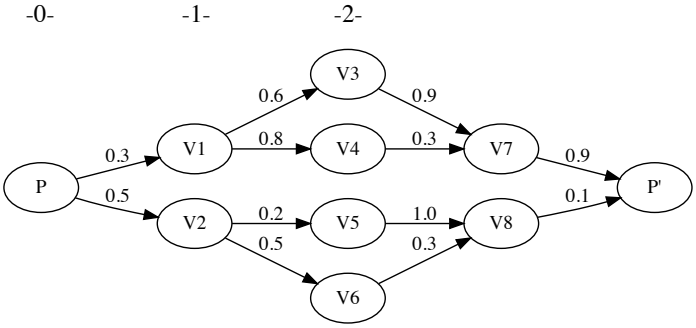

Figure 7.25: Process graph with 2 V-Layers.

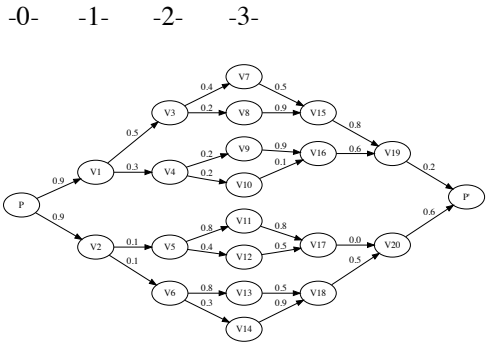

Figure 7.26: Graph with 3 V-Layers.

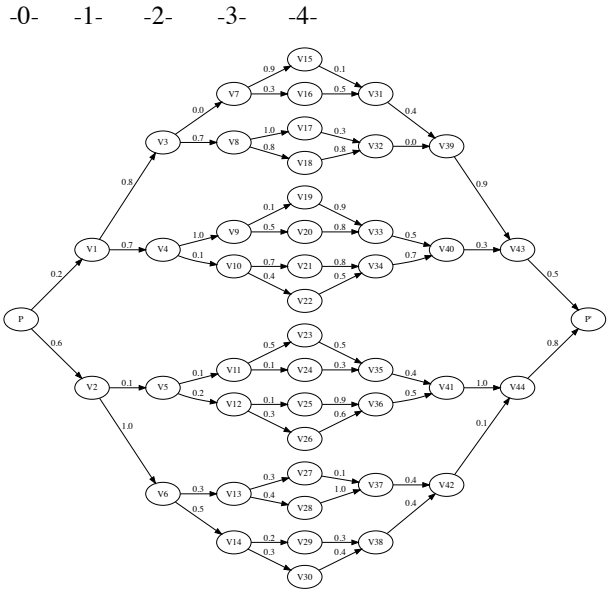

Figure 7.27: Graph with 4 V-Layers.

Table 7.6: Characteristics of generated graphs.

H-Layers	V-Layers	Vertices	Edges	Vertex-to-Edge Ratio
2	1	4	4	1.0
2	2	10	12	1.2
2	3	22	28	1.27
2	4	46	60	1.31
2	5	94	124	1.32
2	6	190	252	1.33
2	7	382	508	1.33
3	3	53	78	1.47
3	4	161	240	1.49
4	3	106	168	1.58
4	4	426	680	1.60

The graph generation process using a continuous H-Layer of 2 is illustrated in Figures 7.24 to 7.27. Increasing the number of V-Layers quickly increases the number of decision points in the graph. The graphs generated in this evaluation include up to 7 V-Layers. However, the majority of business processes is expected to use up to 5 V-Layers. Table 7.6 illustrates the number of vertices, the number of edges, and the vertex-to-edge ratio in graphs with different H-Layers and V-Layers.

7.2.3 Performance Evaluation

The performance requirements are identified by measuring the time of a path-search operation. Time frames required to create new graph objects and to allocate memory for them are excluded for reasons of comparability. The H-Layer is configured to a value of 2, resulting in two outgoing edges of every splitting vertex. The number of V-Layers varies throughout the performance analysis. The JGraphT-implementations of the SPF algorithms Dijkstra, Bellman-Ford, and A* (labeled as AStarE and AStarS in the charts) as well as an all-paths algorithm based on Dijkstra are part of the evaluation.

The computation times for a path search from process start to end are depicted in Figures 7.28 to 7.29. The charts are based on 1000 repetitions per V-Layer and algorithm, the lines represent the mean values of these repetitions. The evaluation indicates the lowest computation times for AStarE, followed by Dijkstra, Bellman-Ford, and AStarS. The performance difference between AStarE and AStarS illustrates the relevance of the applied heuristic, used as landmarks for path calculation (cf. [79]): While AStarE uses the process end vertex as its heuristic, AStarS applies the process start vertex as its heuristic. AStarS performs comparably to Dijkstra and Bellman-Ford

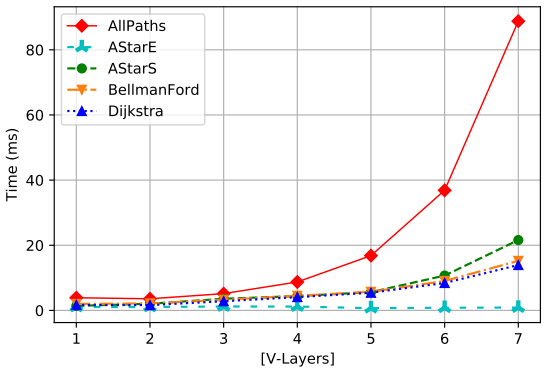

Figure 7.28: Path computation times.

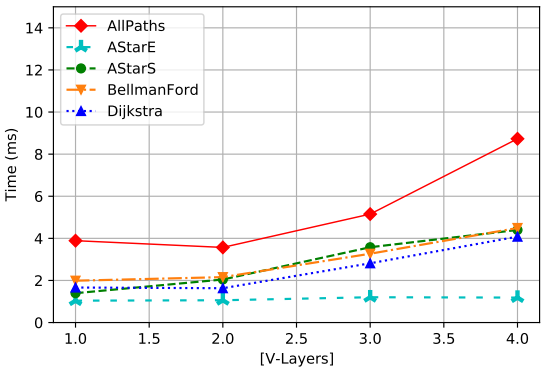

Figure 7.29: Computation times close-up.

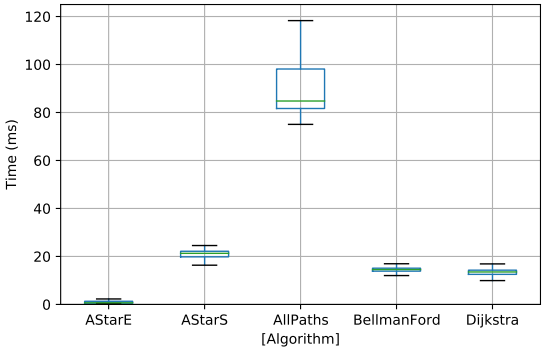

Figure 7.30: Distribution of computation times at 7 V-Layers.

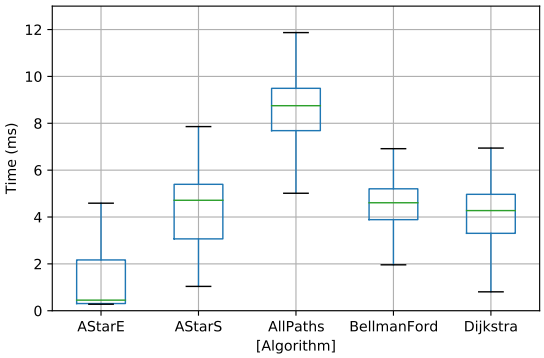

Figure 7.31: Distribution of computation times at 4 V-Layers.

for up to 5 V-Layers, and deteriorates afterward. In contrast, AStarE is identified as the best-performing algorithm (configuration) at all V-Layers.

The highest computation time is consumed by the all-paths algorithm since it identifies all available paths from start to end. However, the difference to SPF calculations only starts to increase heavily at 5 to 7 V-Layers. At V-Layers of 1 to 4, all-paths calculations end after up to 10 ms on average.

A closer look at the distribution of the computation times is provided by Figures 7.30 and 7.31. Each box encapsulates 50 percent of the measured time frame values, the median is depicted by a (green) line within each box. The whiskers on top and at the bottom of a box show the distribution of the remaining values according to [134], outliers are omitted.

The boxplot in Figure 7.30 illustrates computation times at 7 V-Layers. Times for the all-paths algorithm show a distribution around 75 to 120 ms. The SPF algorithms Dijkstra, Bellman-Ford, and AStarS consume around 10 to 30 ms for path computation, while AStarE requires considerably less time. At 4 V-Layers, the all-paths algorithm

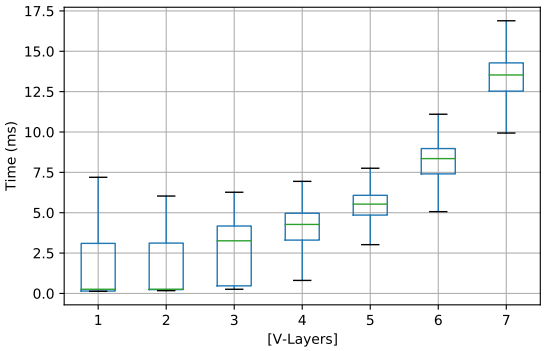

Figure 7.32: Distribution of computation times for Dijkstra.

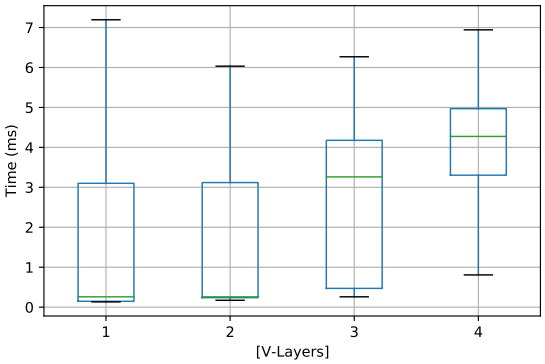

Figure 7.33: Close-up of computation time distribution for Dijkstra.

can operate in a closer range to the SPF algorithms (cf. Figure 7.31). Computation times stay below 12 ms.

Figure 7.32 depicts computation times for Dijkstra at V-Layers 1 to 7. While computation times are about equal for graphs with up to 2 V-Layers (cf. Figure 7.33), the effort increases consistently with growing V-Layers. However, computation times stay below 18 ms at all V-Layers in the evaluation.

The corresponding Cumulative Distribution Functions (CDF) and Kernel Density Estimations (KDE) for V-Layers 7 and 4 are illustrated in Figures 7.34 to 7.37. AStarE outperforms all other algorithms by showing a high density at low computation times. The additional effort carried out by the all-paths algorithm is visible at 4 V-Layers and increasing at 7 V-Layers.

Summarizing the results for the computation time evaluation of the different algorithms indicates minor differences at V-Layers 1 to 4. At V-Layers 5 to 7, the gap

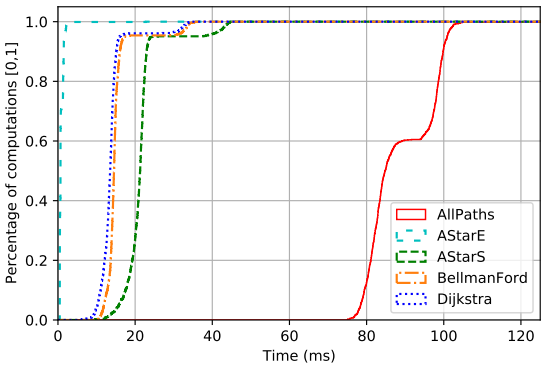

Figure 7.34: CDF at 7 V-Layers.

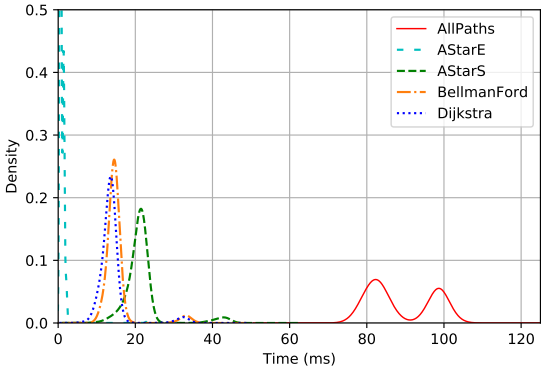

Figure 7.35: KDE at 7 V-Layers.

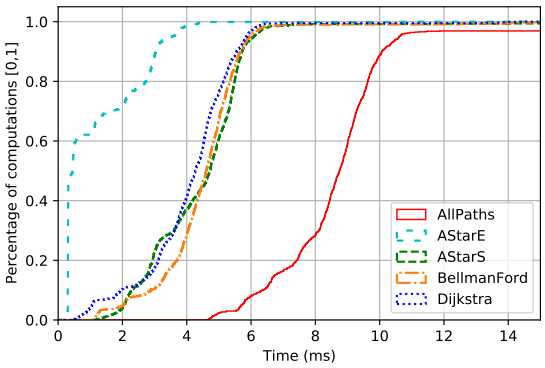

Figure 7.36: CDF at 4 V-Layers.

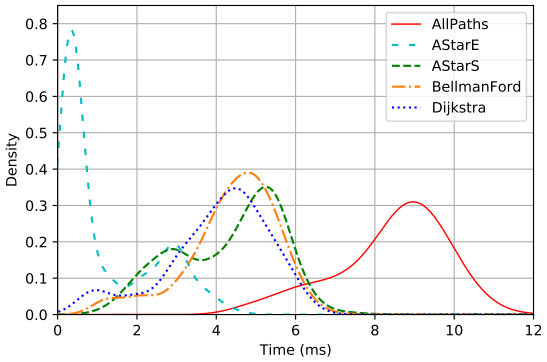

Figure 7.37: KDE at 4 V-Layers.

between SPF and all-paths algorithms grows increasingly. No increase of path calculation times can be identified for AStarE, even at 7 V-Layers.

7.2.4 Resilience Analysis

SPF algorithms calculate costs by summarizing the edge weights of a path. As described in section 4.3.1 (p. 85f.), this may not be optimal for a resilience analysis. A path may be chosen which includes non-resilient edges. Hence, it seems to be reasonable to remove non-resilient edges before applying SPF algorithms on a resilience graph.

While all-paths algorithms can find the most resilient path by comparing metrics of all paths with each other, the question arises whether or not the path identified by an SPF algorithm is a considerable alternative. This subsection evaluates the use of different metrics in conjunction with an SPF and an all-paths algorithm on random graphs. The objective is to investigate suitable combinations of metrics and algorithms. Subsequently, the edge weights configured by the process graph generator are inverted to comply with SPF algorithms (searching for the lowest total path weight R_t). Accordingly, lower edge weight values have a better resilience than higher values. Up to which value an edge weight is considered as resilient is stated in the evaluation scenarios, if relevant.

The all-paths algorithm returns a list of possible graph paths between start and end of a process. Using this list, a path with an improved path level R_l compared to the path level provided by an SPF analysis may be identified. This leads to a more resilient path choice. Due to the inversion of weights, R_l is also inverted, representing the maximum (not minimum) edge weight of a path in this evaluation.

Figure 7.38 compares the path level R_l of an all-paths and a Dijkstra SPF analysis. While the total path weight R_t is lower ($\Rightarrow$ better) for Dijkstra, the path level R_l is lower ($\Rightarrow$ better) for the all-paths analysis. The boxplot in Figure 7.39 illustrates the

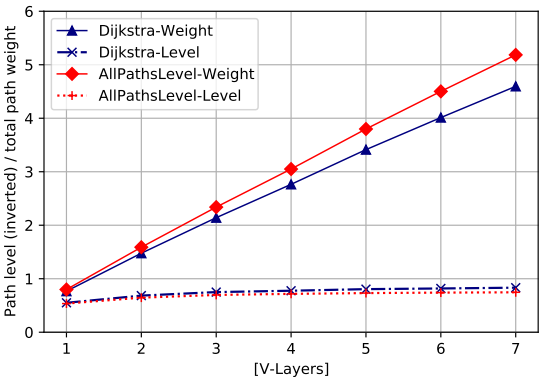

Figure 7.38: Optimizing the inverted path level R_l.

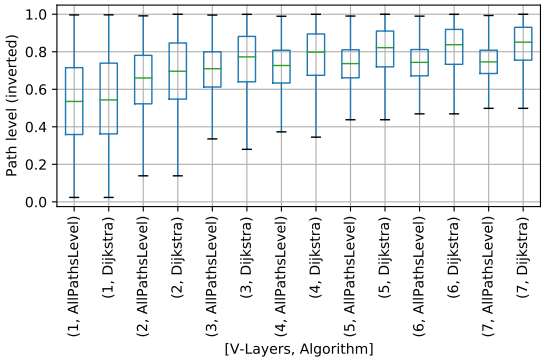

Figure 7.39: Distribution of the inverted path level R_l.

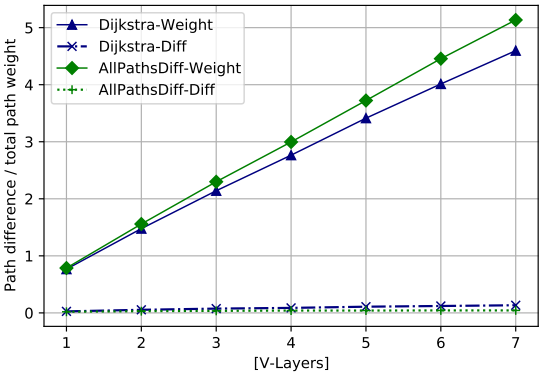

Figure 7.40: Path difference R_d.

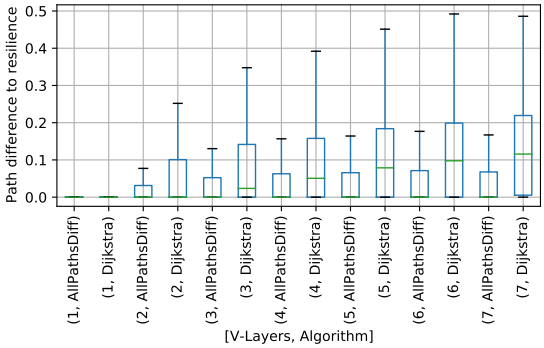

Figure 7.41: Distribution of R_d.

path level distribution for V-Layers 1 to 7. The all-paths-variant can optimize the path level within limits. Depending on the resilient edge definition of a scenario, this may result in a resilient all-paths choice against a non-resilient SPF choice.

Another resilience optimization approach is to minimize the path difference to resilient edges R_d. For the evaluation in Figures 7.40 and 7.41, edges are defined as resilient for $R_e \in \mathbb{R}|0 \leq R_e \leq 0.75$. The charts indicate an optimized path difference compared to the SPF algorithm, especially at high V-Layers. The median in the boxplot of Figure 7.41 remains at 0 for the all-paths analysis, which maps to at least 50 percent of resilient paths. This is a considerable improvement compared to the SPF outcomes.

Besides selecting paths from an all-paths analysis list based on chosen metrics, SPF algorithms can be tweaked to increase resilience. By deleting all non-resilient edges from a graph before an SPF analysis, the resulting path will represent a resilient

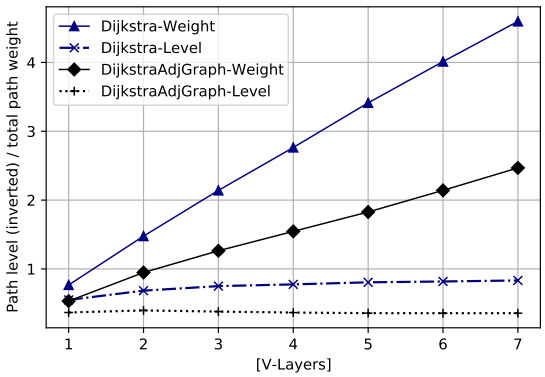

Figure 7.42: Dijkstra operating on unmodified and on adjusted graphs.

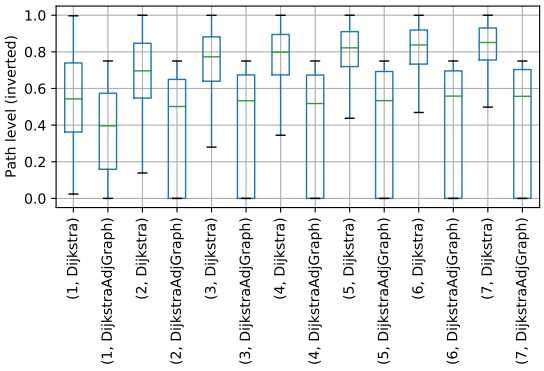

Figure 7.43: R_l-Distribution of Dijkstra variants.

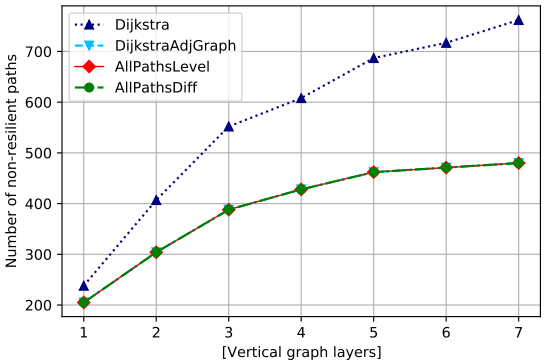

Figure 7.44: Path analysis with resilient edges defined as $R_e \in \mathbb{R} | 0 \leq R_e \leq 0.75$

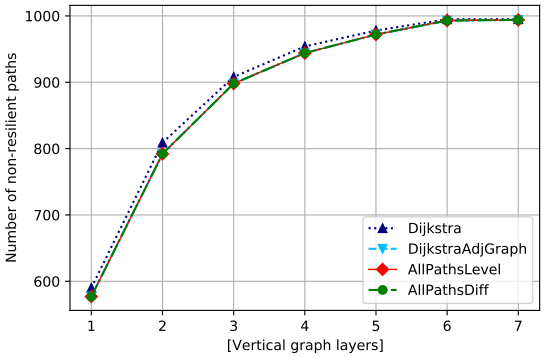

Figure 7.45: Path analysis with resilient edges defined as $R_e \in \mathbb{R} | 0 \leq R_e \leq 0.5$

configuration. However, there may not necessarily be a path in the adjusted graph anymore.

Figures 7.42 and 7.43 illustrate the total path weights R_t and path levels R_l for Dijkstra operating on an adjusted graph free of non-resilient edges (*DijkstraAdjGraph*) and operating on an unmodified graph (*Dijkstra*). Again, a resilient edge is configured as $R_e \in \mathbb{R} | 0 \leq R_e \leq 0.75$. The charts indicate improvements for R_t and R_l.

The most relevant aspect to compare the different approaches is to measure the resilience of the identified paths. Resilient edge values of $R_e \in \mathbb{R} | 0 \leq R_e \leq 0.75$ and $R_e \in \mathbb{R} | 0 \leq R_e \leq 0.5$ have been chosen for the evaluation. The number of non-resilient paths selected by the algorithms within a test out of 1000 runs is depicted in Figures 7.44 and 7.45. The lines of a Dijkstra analysis based on an adjusted graph and two all-paths analyses (based on path level R_l and path difference R_d) are compared to a Dijkstra analysis on the original graph. In both charts, the all-paths analysis and the Dijkstra adjusted graph analysis map on the same line. Results show a significant improvement

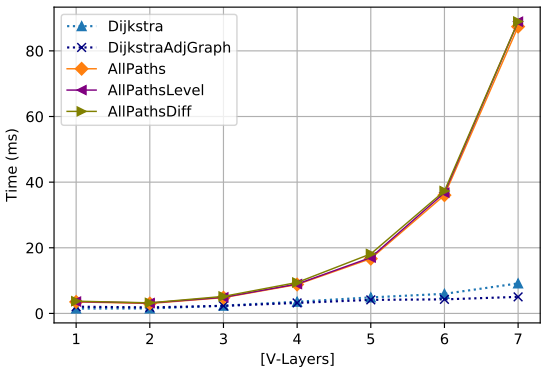

Figure 7.46: Path computation times.

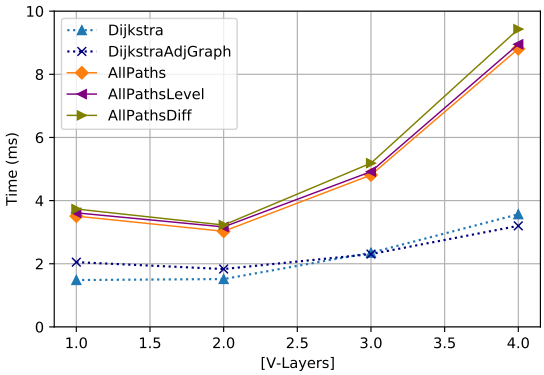

Figure 7.47: Close-up of comp. times.

in reducing the number of non-resilient paths. However, the improvements are minor when using a resilient edge value of 0.5.

Since the comparison between the optimized analysis variants shows no difference in terms of their resilience, it is unclear which analysis should be preferred. A comparison of the computation times of the optimized analysis methods with their original Dijkstra and all-paths variants is depicted in Figures 7.46 and 7.47. The results indicate a slightly larger computation time for the adjusted Dijkstra analysis compared to Dijkstra on an unmodified graph. This is due to the additional effort for removing non-resilient edges from the graph. However, this is only valid for up to 4 V-Layers. At higher V-Layers, the adjusted variant saves computation time since many possible, but non-resilient paths have been already removed by deleting corresponding edges. The additional computation effort for the two all-paths-based variants compared to the original all-paths analysis is minor. A difference between all-paths and all-paths-level can be hardly identified in Figures 7.46 and 7.47.

7.2.5 Scalability Evaluation

The scalability of using graph-based search algorithms to find resilient paths in process graphs is indicated by the Figures presenting the computation times at different V-Layers in section 7.2.3. This section continues the evaluation by further increasing the graph size and adding a second, more powerful computation device.

In addition to the evaluations based on an H-Layer of 2, analyses for H-Layers of 3 and 4 have been added. An Apple MacBook Pro (Early 2015) with a 3.1 GHz Dual-Core Intel Core i7 processor and 16 GB of RAM acts as a second device. The Java Runtime Environment is configured to use 0.5 GB of RAM ($\Rightarrow$ maximum heap size).

The charts for the extended analysis on the Raspberry Pi Zero WH are depicted in Figures 7.48 and 7.49. The Raspberry is challenged at a growing number of vertices

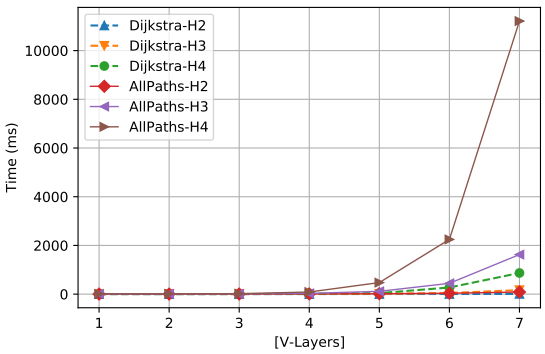

Figure 7.48: Computation times on Raspberry Pi Zero WH.

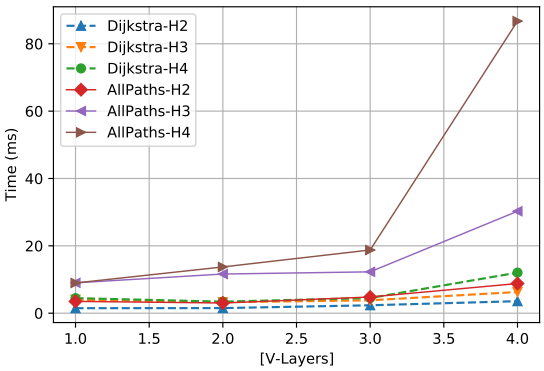

Figure 7.49: Close-up of computation times on Raspberry Pi Zero WH.

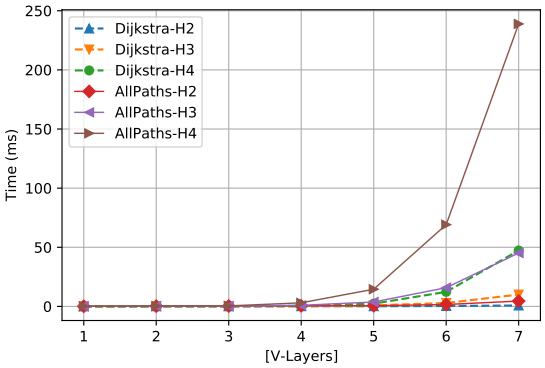

Figure 7.50: Computation times on MacBook Pro.

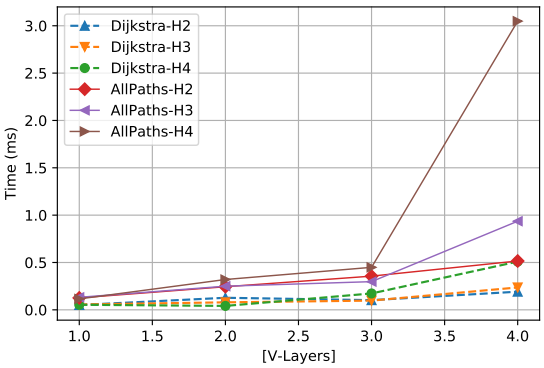

Figure 7.51: Close-up of computation times on MacBook Pro.

and edges. Dijkstra SPF computations may take up to 1 s on average, while all-paths computations may require up to 12 s.

The performance of the MacBook Pro is illustrated in Figures 7.50 and 7.51. Computations for all-paths may take up to 250 ms on average for an H-Layer of 4. However, the results indicate that the MacBook Pro is not challenged by graphs featuring high amounts of vertices and edges. This is especially the case for SPF algorithms.

7.2.6 Findings and Remarks

The results of the evaluation executed on a Raspberry Pi Zero WH indicate no performance and scalability issues when computing resilient paths in typical process graphs. The computation effort for an all-paths algorithm is significantly higher compared to SPF algorithms, especially in large graphs. However, the required average time frame of about 90 ms for an all-paths calculation at V-Layer 7 is comparatively small. In

particular, this is true if a process employs full-feature BPM runtime engines which usually show higher performance demands. The extended evaluation with even larger graphs on a Raspberry Pi Zero WH and a MacBook Pro demonstrates the scalability of graph-based analyses. If computation time is not highly critical, a low-performance device such as the Raspberry may be used. In other scenarios, a more powerful device is suggested to speed up computation time.

The two different A* configurations AStarE and AStarS outline the importance of choosing a reasonable heuristic. While AStarE outperforms Dijkstra and Bellman-Ford at all V-Layers, AStarS performs worse than these two SPF variants at 6 and 7 V-Layers. Since the differences between the SPF algorithms can be ignored in some scenarios, selecting an algorithm may be based on available implementations and implementation effort.

In many scenarios, the majority of process graphs will not exceed the number of 46 vertices / 60 edges (i.e., 2 H-Layers / 4 V-Layers). At least, this was determined for the OPeRAte research project [132]. OPeRAte orchestrates process chains in the area of agriculture, where multiple participants cooperate in unreliable communication environments (i.e., on farms and fields, cf. [72]). The agricultural process chains including slurry applications and maize harvest scenarios typically range between V-Layers 2 to 4 with an H-Layer of 2. Here, a Raspberry Pi Zero WH would completely satisfy the performance requirements.

Computations for a resilient process path may be repeated multiple times during process runtime. Connectivity conditions may change over time, resulting in changed resilience values / corresponding edge weights. Scenarios including a high variability or new processes missing solid statistics are prone to recalculations. This should be considered when planning performance requirements.

A more critical performance consideration of graph algorithms should be done when devices less powerful than a Raspberry Pi Zero WH are used. When using sensors, actuators, or microcontrollers, computation times may be significantly larger with impacts on process operation.

The evaluation of process resilience with defined values for resilient edge weights outlines a problem of SPF algorithms. The traditional way of minimizing cost may include non-resilient edges in the path, resulting in failing processes. Since all SPF algorithms follow the same objective, there is no difference in the chosen process path among them.

7.3 Recommendations for Process Modeling and Execution

The previous sections evaluated the concepts and approaches for the modeling and execution of resilient processes presented in this thesis. This section interprets the results and their meanings for the modeling and execution in unreliable communication environments. Recommendations for the modeling, analysis, and execution of processes are provided.

7.3.1 Process Modeling

The abstract modeling examples used throughout this thesis' chapters 3 to 6 (p. 41ff.) as well as the water heating scenario (cf. Figure 3.5, p. 51), the disaster relief scenario (cf. Figure 3.6, p. 52) and the environmental-friendly agricultural slurry application demonstrate the versatility and flexibility of *rBPMN's* modeling concepts. Using these concepts, the existing slurry process model of section 2.2.4 (p. 25ff.) has been identified as non-resilient (cf. section 7.1). Modifying the original slurry process *S1* resulted in a resilient process model *S3*, suitable for slurry application scenarios with varying demands regarding connectivity and optimal process operation.

The addition of alternatives is beneficial not only for ensuring resilient operation but for the multi-criteria-driven selection of the best-suited process path. Domain experts are enabled to explicitly state their choice in the selection of alternatives by using *OppPriorityFlows*. Decisions based on characteristics of alternatives are facilitated by *OppDecisionFlows*. The integration of dynamically appearing participants enlarges the opportunities for resilient operation at runtime. Failing of connectivity is compensated by the movement of functionality across participants. Table 7.7 provides recommendations for the use of modeling concepts considering different aspects.

A key aspect in process modeling is to continuously update and analyze the process model regarding resilient and optimal operation. Individual scenario characteristics have a big impact on resilience and optimal operation, which makes their integration essential for the process analysis. Gathering information about connectivity, available participants, and services should be automated and logged during process execution. Using these logs, the process model may be re-evaluated for resilient and optimal operation. Potential limitations of the model may be addressed, resulting in improvements for prospective process executions. This leads to the lifecycle of process models illustrated in Figure 7.52. As depicted, a process model is continuously being optimized for future use.

Table 7.7: Recommendations for modeling resilient processes.

Modeling aspect	**Recommended modeling concept**
Possibility of insufficient connectivity.	**Option 1:** Add alternatives using *OppPriorityFlows / OppDecisionFlows*. **Option 2:** Enable dynamic alternatives at runtime (*OppDynTask*). **Option 3:** Move functionality locally (*MovTask / MovSubProcess / MovParticipant* and *OppTask*).
Possibility of no connectivity.	Move functionality locally (*MovTask / MovSubProcess / MovParticipant* and *OppTask*).
Decision-making based on explicit priorities defined by domain experts.	*OppPriorityFlows*.
Decision-making based on characteristics of alternatives.	*OppDecisionFlows*.
Provisioning of functionality for other participants.	Declare a *MovTask / MovSubProcess / MovParticipant*.
Task shall be able to execute functionality locally.	*OppTask / OppDynTask*.
Integration of process segment offered by participant is optional.	Optional *OppMessageFlow*.

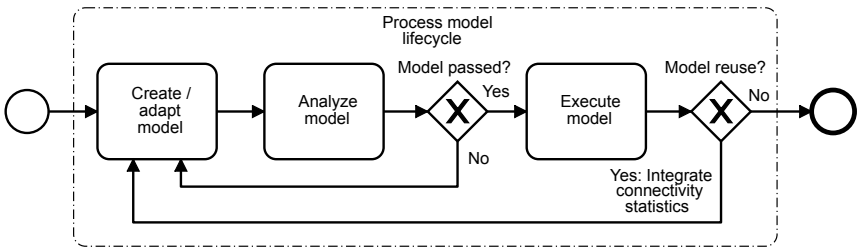

Figure 7.52: The lifecycle of a process model (modeled in BPMN).

7.3.2 Resilience Analysis

If resilient process paths shall be found and ranked, connectivity characteristics may be preferred against connectivity probabilities as edge weights. Connectivity characteristics allow to differentiate and rank resilient process paths. In contrast, connectivity probabilities only identify two resilient paths with a path probability of $P_p = 1$, unable to prioritize the more resilient path.

Likewise, integrating a connectivity safety margin to compensate differences between estimations and real-world connectivity requires the use of connectivity characteristics. However, connectivity characteristic edge weights require detailed knowledge or statistics about the scenario's connectivity. If limited knowledge or only simple connectivity statistics exist, connectivity probabilities might be a reasonable choice as edge weights. Furthermore, connectivity probability edge weights are not subjected to over-weighting and are suitable for a combined-path analysis. Summarized recommendations for the usage of connectivity characteristics and connectivity probabilities are provided in Table 7.8.

The investigation of graph-based search algorithms indicates that the application of SPF/LPF algorithms may not be optimal for every scenario due to their focus on the total resilience R_t. A promising choice is an all-paths analysis combined with a use-case-driven selection of appropriate resilience metrics. Most scenarios may operate well by focusing on a preferably high resilience level of the path R_l or a high path probability P_p. If the accuracy of edge weights is uncertain or weight values change rapidly due to scenario-related circumstances, utilization of the maximum-step heuristic is considerable. Finally, in scenarios featuring different communication technologies, a combined-path analysis may optimize resilience by combining process paths. A graph preparation (e.g., limiting maximum edge weights, removing edges not satisfying the

Table 7.8: Recommendations for the use of connectivity characteristics and connectivity probabilities as graph edge weights.

Scenario connectivity condition	**Recommended edge weight type**
Connectivity statics are comprehensive / originate from the same or a comparable scenario.	Detailed analysis results able to be ranked using connectivity characteristics. Solid and simplified results using connectivity probabilities.
Connectivity statics are limited / originate from other scenario.	Use connectivity characteristics and add a connectivity safety margin for resilience. Use connectivity probabilities with uncertainty in mind.
Connectivity statics do not exist.	Try to estimate connectivity. Connectivity probabilities require less technical details for the resilience calculations.

Table 7.9: Recommendations for the resilience analysis.

Analysis aspect	**Recommended analysis approach**
Domain expert is unfamiliar with scenario / scenario is novel.	Comparison-based analysis (all-paths) at design time to identify scenario characteristics and behavior.
General approach for non-specific scenarios, identifying the most resilient path.	Comparison-based analysis (all-paths), choosing the path with the highest resilience level R_l.
General approach for performance- or time-restricted scenarios, choosing a path with solid resilience.	Comparison-based analysis (all-paths) at design time to identify a minimum required resilience level R_l for the scenario. At runtime: Remove all edges not meeting the minimum R_l from the graph and perform an SPF/LPF search to select a path.
Scenario is highly dynamic / path decisions are only of local importance.	Maximum-step analysis as a performance-optimized alternative.
Challenging scenario in terms of connectivity, multiple types of communication technologies in use.	Combined-paths analysis may establish/improve resilience.

required resilience) has shown to be useful before applying graph-based search algorithms for the analysis. Recommendations for the resilience analysis are summarized in Table 7.9.

7.3.3 Multi-Criteria Analysis

Categorization and prioritization of criteria relevant for a process are guided by the concept of three different importance levels (cf. Table 5.2, p. 98). In many cases, a criterion may have a hard threshold and an optimization direction at the same time. Both may be respected by setting a first level hard threshold and adding a second level optimization statement. An example is provided in section 7.1.2: For the criteria accuracy and cost, hard thresholds as well as optimization directions have to be respected (cf. Table 7.4, p. 154). Recommendations are summarized in Table 7.10.

Depending on the criteria set, meanings of criteria edge weights may differentiate significantly from each other. Where possible, weight normalization to a consistent scale (e.g., $C_w^x \in [0, 1]$) increases comparability across different criteria. Discrete and joint criteria graphs allow the definition of edge weights in a flexible way. Joint graphs combine graphs for criteria that share a common understanding of edge weights (e.g., cost and time). If criteria are diverse in their definition (e.g., value meaning, value range, related edges, and vertices) and can not / shall not be normalized, discrete graphs for each criterion are appropriate.

Table 7.10: Recommendations for the categorization and prioritization of a multi-criteria set.

Criteria aspect	**Recommended categorization**
Hard threshold for criterion.	Define as of 1. level importance, state threshold.
Optimizing direction for criterion (e.g., maximum, minimum).	Define as of 2. level importance, state optimization direction.
Hard threshold and optimization requirement for criterion.	Define as of 1. level importance and state threshold. Additionally define as of 2. level importance and state optimization direction.

The selection of appropriate multi-criteria metrics heavily depends on the applied criteria. For measuring resilience, the resilience level of the path C_l^R is a reasonable metric. Other criteria such as cost and time may find the total path weight C_t^x and other economical metrics such as C_a^x, C_l^x, and C_m^x helpful. For other criteria such as the accuracy, privacy, and automation of process operation, the appropriate metrics are less explicit and depend on the concrete scenario.

Recommendations for using the different approaches for the multi-criteria process analysis are provided in Table 7.11. In general, it is beneficial identifying the scenario characteristics by performing a comparison-based analysis at design time. Domain experts may compare the chosen metrics, optimize the model and re-evaluate the results. Radar charts may help to illustrate scenario characteristics and to identify appropriate metrics. For choosing the most appropriate path at runtime, many scenarios may continue to apply a comparison-based analysis using an all-paths search in combination with one or more metrics ranking the remaining paths for selection.

Besides, the design-time results may help for setting up an iteration-based analysis procedure. This is especially helpful in scenarios limited regarding computation time and performance. An all-paths or an SPF/LPF algorithm may be used on the final graph to select a path. An iteration-based analysis procedure is inadequate for criteria metrics based on summarized edge weights (e.g., total path weight C_t^x). Here, it may be an alternative searching for KSPs, filter the paths list according to the criteria metrics and select the remaining, highest-ranked path.

The utilization of multi-criteria graph algorithms should be considered carefully. Many algorithms identify a set of Pareto-optimal paths. However, the evaluation of the slurry process example in section 7.1 showed that the Pareto-optimal paths have to be filtered for reasonable process paths. Other algorithms use scalarization techniques, which prevent the investigation of distinct metrics for individual criteria. A chosen path may include edges unqualified for some criteria. Also, many multi-criteria algorithms

Table 7.11: Recommendations for the multi-criteria analysis.

Analysis aspect	**Recommended analysis approach**
Domain expert is unfamiliar with scenario / scenario is novel.	Comparison-based analysis (all-paths) at design time to identify scenario characteristics and behavior.
Hard thresholds for criteria exist.	Iteration-based analysis, graph-search on final graph (SPF/LPF, all-paths).
Final graph: Chosen metric calculates total path weight.	SPF for finding total minimum / LPF for finding total maximum.
Final graph: Chosen metric identifies an aspect of a path (e.g., minimum / average / maximum edge weight).	All-paths search.
All criteria / a criteria subset aims at minimizing / maximizing a summarized metric.	Use a (weighted) joint graph to combine the corresponding criteria. The joint graph may be used for an SPF/LPF graph search.
Criteria set and its requirements are diverse and do not fit an iteration- / comparison-based analysis.	Develop a scenario-based analysis by combining separate / joint criteria graphs and iteration- / comparison-based analysis approaches.

employ the SPF concept. However, this may be challenging if a criteria set is not aiming at minimizing a summarized metric, such as cost.

7.3.4 Process Execution

An *rBPMN* runtime engine has not been implemented in this thesis and existing BPMN runtime engines are unaware of *rBPMN's* modeling elements. Hence, additional effort is required for the execution of *rBPMN*-based processes. The required implementation effort depends on the used *rBPMN* elements in relevant scenarios. Reducing effort is possible by only implementing the parts of *rBPMN* that are used by the applied scenarios. Table 7.12 summarizes recommendations regarding the implementation when executing *rBPMN*-based processes.

Two approaches for adding *rBPMN* support are the *i)* adaptation of an existing BPMN runtime engine and the *ii)* integration of *rBPMN's* resilience strategies into linked implementation code of process activities. While some codebases of runtime engines are open source, they may be closed source in other products. Technical opportunities for custom code implementations depend on the chosen runtime engine. The proof-of-concept implements custom code for the discovery of and decision-making on alternatives as part of Camunda process modules. Reuse of the code is straightforward by extracting the corresponding classes into a library, available for the use in other process implementations. While this approach minimizes the effort for implementation, the proof-of-concept also illustrated drawbacks of mapping *rBPMN* process models to

Table 7.12: Recommendations for the scenario-driven implementation of *rBPMN's* modeling elements and analysis approaches.

Scenario requirements	**Recommendation**
Scenario requires to identify/verify resilient process operation.	Use *OppMessageFlows* with related QoS, message and connectivity properties and the resilience analysis approach of *rBPMN*.
Scenario requires to identify/verify and to establish/optimize resilient process operation.	Use the full feature set (communication-, collaboration and decision-related parts) of *rBPMN* in combination with its resilience analysis approach.
Scenario requires to identify/verify and establish/optimize process operation driven by resilience and other criteria.	Use the full feature set of *rBPMN* in combination with its multi-criteria analysis approaches.

BPMN models compatible with Camunda. By removing visual attributes for alternative groups and locally moved functionality, the resulting model gives a false impression about the process workflow. Resilience verification is no longer possible since information about QoS, message properties, and connectivity properties is lost. Alternative groups and their selection criteria (priority-based, characteristics-based) have to be moved into the delegate classes of activities.

Integration of state-of-the-art technologies like microservices supports the implementation of *rBPMN's* resilience strategies in real-world environments. Comprehensive software frameworks may be part of the process execution environment. While frameworks such as Spring facilitate the development of code used in processes, some functionality has to be used with caution in unreliable communication environments. For instance, using the circuit-breaker pattern [71] helps to avoid overload situations in cloud environments. However, it may falsely prevent access to services and participants in dynamic, intermittent scenarios taking place in unreliable communication environments.

Adaptation effort for existing microservices is reasonable. The implemented methods for neighbor and service discovery as well as for decision-making need to be integrated. While *Spring Boot*, *Spring Eureka*, and *Camunda BPM* have been used in the evaluation, other technologies such as *Signavio, jBPM, docker, rkt, Zookeeper*, and *Consul* may be used as BPM runtime engines, to implement and move microservices and to discover services. Alternatively, integration of tools like the Spring Framework with BPMN runtime engines like Camunda and the *rBPMN* metamodel may simplify the effort for resilient process execution.

Decision-making on alternatives is realized in a highly dynamic manner by identifying, comparing, rating, and selecting alternatives at runtime. A lesson learned at this

point of implementation is the need to check service availability information gathered from Eureka service registries. Depending on the Eureka configuration, services may be shown as available while they have already disappeared. Connectivity checks prior to service utilization are used in the proof-of-concept to rapidly exclude unavailable services and to decide on a new alternative (cf. Figure 6.7, p. 136). Instead of querying and combining service information of different Eureka servers, a single query to the local Eureka server may identify all available services by using Eurekas replica mechanism. Services of other Eureka servers become part of the local server instance. While this principle facilitates service discovery, replica configuration of servers in highly dynamic scenarios may be challenging.

The configuration of network settings has a direct influence on the resilience strategies of *rBPMN*. For instance, frequent *hello* broadcast messages (e.g., every five seconds) may allow recognizing moving participants rapidly in a scenario. Otherwise, a less frequent interval of hello broadcasts may prevent identification and usage of neighboring participants and their offered services.

Summary

This chapter evaluates *rBPMN's* concepts and approaches for the modeling and execution of resilient processes in unreliable communication environments. Using *rBPMN's* modeling elements, a resilient and optimal operating process for the exemplary agricultural slurry scenarios can be achieved and verified. Proof-of-concept implementations demonstrate that *rBPMN's* resiliency strategies and analysis approaches can be combined with state-of-the-art design principles and technologies such as microservices and the service discovery provided by the Spring framework. The multi-criteria analysis of the slurry scenario illustrates the importance of flexible analysis approaches, which are able to be customized for the corresponding scenario. No performance and scalability issues of graph-based analyses have been identified during evaluation on limited hardware for representative graph sizes. The chapter concludes by providing recommendations for the modeling, analysis, and execution of resilient processes using *rBPMN*.

CHAPTER 8

CONCLUSION

This thesis presents concepts and approaches for the modeling and execution of resilient business processes in unreliable communication environments. The addressed challenges, contributions, and findings are summarized subsequently. The chapter concludes with an outlook regarding directions of future work.

8.1 Summary of Challenges, Contributions, and Findings

The Business Process Model and Notation (BPMN) is a widespread Process Modeling Language (PML) used in many different application domains and scenarios. It allows the graphical modeling of collaborative workflows including participants of different organizations. By linking process models with activity-related source code, BPMN runtime engines can execute, monitor, control, and document business processes.

Unreliable connectivity challenges the execution of business processes. The operation of processes not designed for delayed, intermittent, or broken connectivity quickly fails. In the context of BPMN, this leads to the following problem statement.

Problem Statement

Preparing BPMN process models for unreliable connectivity is challenging. In many cases, the integration of alternatives for possibly failing message flows is limited, cumbersome, and inflexible. While alternatives may be defined for specific scenario conditions using gateways, error events, and business rule tasks, a flexible integration of

alternatives for scenarios with varying objectives based on the same process model often fails. Adapted models often miss the focus on the actual process objectives due to a cumbersome, time- and space-consuming modeling of alternatives.

Process models are missing a resilience verification mechanism, preventing failures and breakdowns of operation during process execution. By verifying communication resilience at design time, domain experts may identify and optimize potentially failing process segments before execution. Since the resilient operation is a major, but not the only criterion of interest for most processes, an analysis approach to optimize operation regarding a diverse set of criteria is missing.

While a resilient process model is a requirement, its sole existence is not preventing process failures. Aspects such as the initial process configuration and the movement of functionality before runtime, the dynamic identification of neighboring participants as well as the discovery and usage of offered services as alternatives for breaking message flows require a cross-layer information exchange between the network and the application layer of the ISO/OSI model. Execution strategies are missing to maintain resilient process operation.

Based on these findings, this thesis addresses four main challenges for collaborative processes taking place in unreliable communication environments. The findings and contributions are summarized subsequently.

Modeling of Resilient Processes (Challenge 1)

With *resilient BPMN* (*rBPMN*), this thesis introduces a BPMN metamodel extension for the modeling of resilient business processes exposed to unreliable communication.

The extension concepts of *rBPMN* allow domain experts to adapt existing BPMN process models for unreliable communication environments. By integrating *opportunistic message flows (OppMessageFlows)*, communication with other participants is characterized as potentially delayed, intermittent, or broken. Connectivity issues with other participants can be addressed by *adding alternatives*. Domain experts may model sets of alternatives and configure decision-making based on static priorities or on characteristics of the alternatives. Also, services offered by neighboring participants can be identified as additional alternatives not considered at process design time. This lowers the risk of process interruptions and breakdowns in unreliable environments.

Movable functionality enables participants to locally move and execute the functionality of other participants. A process remains capable to operate even if there is no connectivity to any collaborative participant.

Decision-making on alternatives is not limited to the consideration of communication resilience. Domain experts may include other criteria such as accuracy, cost, and

time of process activities along with resilience. This way, the decision-making procedure can dynamically select the optimal process path under the given circumstances.

Process Resilience Verification (Challenge 2)

An important aspect is the *verification of process resilience* at design time. As a result, domain experts can identify vulnerable process segments and strengthen their resilience to connectivity failures. *rBPMN* integrates *Quality of Service (QoS) requirements* and *scenario-driven connectivity properties*, enabling statements about the resilience of message flows to collaborative participants.

Verifying a complete process model for resilient operation arises the question of how process paths may be verified and what metrics should be used to verify and rank the resilience of different process paths. This thesis introduces a *graph-based approach for the resilience analysis*. Process models are translated into graphs, using the resilience calculations of message flows as graph edge weights. Following, graph algorithms are applied to evaluate metrics of process paths.

Various metrics are available for analyzing different characteristics of a process path. A major metric is the resilience level of a path R_l since it indicates whether or not a path is resilient. Also, R_l seems to be a reasonable choice for identifying/ranking process paths against each other.

A *comparison-based resilience analysis* using an all-paths algorithm lists all available paths with a set of chosen metrics. The metrics allow to filter the path list and select the best-suited path. Other algorithms may be considered to reduce computation efforts regarding performance and time requirements. For instance, an *Shortest-Path-First (SPF) search* may automatically identify and select a graph path.

Multi-Criteria Process Operation (Challenge 3)

With different resilient process paths on hand, the question is how to select the best-suited path. While some scenarios may choose the most resilient path, others may consider additional criteria along with communication resilience. This thesis addresses this issue by extending the graph-based resilience analysis into a *multi-criteria optimization analysis*.

An important aspect of the multi-criteria process-to-graph translation is to respect process segments including repetitions, such as loops. The meaning of repeating segments can differ for every criterion. For instance, repeatedly calling a service requiring a yearly subscription is not increasing the process' costs. However, it may improve the accuracy of a parameter optimized by the service and increase the time needed

for the corresponding process path. This thesis introduces an approach in which every criterion can address or ignore a repetition segment, depending on its demands.

Edge weights are applied to the translated process graph, representing the chosen criteria. The different criteria weights of an edge can be combined into a single weight value. Principles such as the Weighted Sum Model (WSM) allow to weight the criteria against each other according to their importance. This results in a joint criteria graph with combined edge weights. An alternative is the creation of separate criteria graphs. Here, the process graph is duplicated for every criterion and is assigned with the corresponding edge weights.

The capability to design the *multi-criteria graph analysis* in a flexible and customizable manner is of main importance. This guarantees the ability to respect the specific characteristics of every scenario and to observe the chosen criteria metrics. This thesis introduces different approaches for the multi-criteria analysis. The *iteration-based analysis* allows straightforward automatization by removing unqualified graph edges and applying a graph-based search algorithm, such as SPF. A *comparison-based analysis* results in a path list to be filtered using chosen criteria metrics, while an *algorithm-based analysis* allows to directly respect multiple criteria using multi-criteria graph algorithms. The different analysis approaches may be combined in a scenario-driven way to a *scenario-based analysis*.

Resilient Process Execution (Challenge 4)

While a resilient process model includes strategies to maintain resilience during operation, an execution environment needs to implement and support the resilience strategies introduced by the modeling concepts. For example, decision-making on a set of alternatives requires to dynamically identify, discover and use participants and their offered services at runtime. This thesis introduces an approach interfacing with routing information of the network layer to identify neighboring participants. Procedures for the identification and usage of offered functionality are presented. Suggestions for the implementation of movable functionality and its distribution across participants are provided.

Evaluation

The concepts and approaches of *rBPMN* have been implemented and evaluated in a *proof-of-concept*. The multi-criteria optimization of the agricultural slurry scenario demonstrates a high flexibility of the introduced graph-based analysis approaches. While the criteria resilience, accuracy, cost, and time have been chosen for this example, others such as error ratio, self-sufficiency level, data volume, or performance

requirements can be used for other scenarios. This also includes scenarios that do not make use of BPMN or *rBPMN*, where the process-to-graph translation principle of creating separated and extended graph paths remains identical. An extensive evaluation of the graph analysis using processes of different sizes did not identify any performance or scalability problems.

An on-demand identification and usage of service offering participants guarantees to include all available alternatives at process runtime. Dynamically appearing participants originally not part of the process model are integrated as additional alternatives.

The movement of functionality is illustrated by moving process modules of the Camunda runtime engine and self-contained microservices. Profiles simplify microservice configuration to use the same piece of executable code in cloud environments and locally at participants. Other technologies like container virtualization may be used alternatively.

Results

rBPMN with its concepts and approaches for the modeling and execution of resilient processes is able to cope with the challenges faced by business processes taking place in unreliable communication environments. The research objectives defined in section 1.3 (p. 5f.) are successfully addressed by this thesis.

In conclusion, the scientific contributions of this thesis can be summarized as follows:

- The BPMN metamodel extension *rBPMN*, introducing modeling concepts for resilient process models (*Challenge 1*);
- A graph-based resilience verification approach for process models (*Challenge 2*);
- Graph-based multi-criteria analysis approaches, identifying the process path of optimal operation (*Challenge 3*);
- Two proof-of-concept implementations, realizing *rBPMN's* resilience strategies and serving as guides for other application domains (*Challenge 4*);
- Recommendations for the modeling, analysis, and execution of resilient business processes (part of *Challenges 1-4*).

The core results of this thesis have been published in five publications, listed in chapter Publications (p. 219ff.). Further on, the chapter mentions many more publications related to this thesis' problem statement, published by the author of this thesis.

8.2 Directions of Future Work

The focus of this thesis lies in the elaboration of concepts and approaches for the challenges raised. Other aspects for the modeling and execution of processes happening in unreliable communication environments deserve further investigations. Some of these aspects are listed subsequently.

Tools for Process Modeling and Execution

rBPMN adds new elements for resilient process models to the BPMN modeling palette. Analysis approaches allow to verify resilient operation and to optimize a process for a set of diverse criteria. Weak process segments may be identified and optimized before process runtime.

Domain experts would benefit from additional tools for modeling and execution. The integrating of message flow properties, QoS requirements, and scenario-driven connectivity estimations could be eased by supporting tools, analyzing process communication and suggesting technical parameter values. The results of a resilience analysis may be presented visually in the process model, illustrating solid, fragile, and poor segments. Also, a tool could provide recommendations for strengthening resilience after an analysis. The preparation of a collaborative scenario including participant configurations and the movement of functionality between participants could be automized.

An *rBPMN*-capable Process Runtime Environment

This thesis' evaluation translated *rBPMN* models into BPMN models for execution using the Camunda runtime engine. An *rBPMN*-capable runtime could tightly integrate the added modeling elements as well as the resilience and multi-criteria analysis approaches. Further on, custom code linked to activities realizing the service discovery, graph analysis, and path selection may be reduced or avoided by extracting decision-making configurations from process models.

Network-Layer Configuration

The network layer configuration has a direct impact on process execution. Especially in highly mobile and dynamic scenarios, the rapid discovery of neighboring participants is highly important. However, neighboring participants are discovered as part of the network routing. Research on using the best-suited network routing algorithms and their configuration for discovering local neighbors is needed. Concepts such as the custody transfer of packets by intermediate participants may create additional communication possibilities. Specialized types of routing algorithms such as geographical routing ap-

proaches may have a positive influence on connectivity. Routing for opportunistically and seamlessly connected participants in the cloud is needed and has to be combined. Besides, analyzing and integrating the ideal network layer configuration into the BPMN process model may be a reasonable approach.

Cross-Layer Information Exchange

The network layer knows best about the current status of connectivity. By analyzing past connectivity characteristics during process execution, a prediction for future connectivity may be created. Sharing this data with the process being executed as part of the application layer would allow the process to adapt its communication to the predictions of the network layer. The other way around, the network layer may be adapted for the variable needs of the process. For instance, the rapid discovery of neighboring participants is often only required at some points of the process. The hello broadcast of proactive routing algorithms could be adapted driven by the needs of the executed process.

Communication Optimizing Middleware

A middleware placed between network and application layer can be an approach to collect, combine, analyze, and share information to optimize process communication. By sharing connectivity and graph analysis information with other participants, knowledge about network segments out of the coverage range of the local participant can be gained. The local graph-based decision-making may be updated, resulting in more sophisticated decisions for the selection of process paths.

Optimization of Resilience using Process Mining

Statistics about connectivity and available participants of previous process executions are beneficial for the resilience analysis at design time. Further knowledge for the optimization of process models may be gained by using process mining techniques [166] on past process data (event logs).

Optimization of Resilience using Artificial Intelligence

Process characteristics and behavioral patterns of participants may be identified using artificial intelligence, allowing proactive adaptations of process operation at runtime. For instance, the availability of certain service-offering participants may be a correlation of location and time. The process would be able to predict the availability of the corresponding participants.

Graph-based Decision-Making for Business Processes

The graph-based multi-criteria approach of analyzing processes for the best-suited process path may be beneficial for business processes in general. The definition of decision-making based on characteristics of activities adds a new level of flexibility to process models. Implementing an equivalent level of flexibility using existing modeling concepts such as gateways, events, and business rule tasks often fails. Therefore, graph-based process analyses are also beneficial for processes not exposed to unreliable communication.

BIBLIOGRAPHY

[1] I. Abouzid and R. Saidi. Proposal of BPMN Extensions for Modelling Manufacturing Processes. In *2019 5th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE, 2019.

[2] Activiti. Open Source Business Automation, www.activiti.org, last accessed: 2021-10-10.

[3] K. Ahmad, N. I. Udzir, and G. C. Deka. *Opportunistic Networks: Mobility Models, Protocols, Security, and Privacy.* CRC Press, 2018.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Alfred P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1988.

[5] T. Allweyer. *BPMN 2.0: Introduction to the Standard for Business Process Modeling.* Books on Demand, 2016. ISBN 9783837093315.

[6] ANEDO. Hülsmeyerstr. 35, 49406 Eydelstedt, Germany. www.anedo.com, last accessed: 2021-10-10.

[7] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile Ad Hoc Networking: Cutting Edge Directions.* IEEE Series on Digital & Mobile Communication. Wiley, 2013. ISBN 9781118511237.

[8] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route Planning in Transportation Networks. In *Algorithm Engineering*, pages 19–80. Springer, 2016.

[9] R. Bauer and D. Delling. SHARC: Fast and Robust Unidirectional Routing. *Journal of Experimental Algorithmics (JEA)*, 14:2–4, 2010.

[10] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16(1): 87–90, 1958.

[11] H. Betke and M. Seifert. BPMN for Disaster Response Processes. In *Informatik 2017*, pages 1311–1324. Gesellschaft für Informatik, Bonn, 2017.

[12] C. Blum and D. Merkle. *Swarm Intelligence: Introduction and Applications*. Springer, 2008.

[13] P. Bocciarelli and A. D'Ambrogio. A BPMN Extension for Modeling non-functional Properties of Business Processes. In *Proceedings of the 2011 Symposium on Theory of Modeling & Simulation*, pages 160–168. Society for Computer Simulation International, 2011.

[14] P. Bocciarelli and A. D'Ambrogio. A model-driven Method for Enacting the design-time QoS Analysis of Business Processes. *Software & Systems Modeling*, 13(2):573–598, 2014.

[15] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia. Simulation-based Performance and Reliability Analysis of Business Processes. In *Proceedings of the 2014 Winter Simulation Conference*, pages 3012–3023. IEEE Press, 2014.

[16] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia. A BPMN Extension to enable the explicit Modeling of Task Resources. In *Proceedings of the 2nd INCOSE Italia Conference on Systems Engineering (CIISE)*, pages 40–47, 2016.

[17] P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia. A BPMN Extension for Modeling Cyber-Physical-Production-Systems in the Context of Industry 4.0. In *14th International Conference on Networking, Sensing and Control (ICNSC)*, pages 599–604. IEEE, 2017.

[18] R. Braun. Behind the Scenes of the BPMN Extension Mechanism Principles, Problems and Options for Improvement. In *3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 1–8. IEEE, 2015.

[19] R. Braun and W. Esswein. Classification of domain-specific BPMN Extensions. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 42–57. Springer, 2014.

[20] R. Braun and W. Esswein. Entwicklung einer BPMN-Extension für ressourcenintensive Prozesse im Maschinenbau. *Tagungsband Multikonferenz Wirtschaftsinformatik*, 2014:1574–1586, 2014.

[21] R. Braun, H. Schlieter, M. Burwitz, and W. Esswein. BPMN4CP: Design and Implementation of a BPMN Extension for Clinical Pathways. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 9–16. IEEE, 2014.

[22] L. S. Buriol, M. G. Resende, and M. Thorup. Speeding up Dynamic Shortest-Path Algorithms. *INFORMS Journal on Computing*, 20(2):191–204, 2008.

[23] S. Burleigh, K. Fall, and E. Birrane. Bundle Protocol Version 7 (Internet-Draft 25). *IETF*, 2020.

[24] Camunda. BPMN and Microservices Orchestration, Part 1 of 2: Flow Languages, Engines, and Timeless Patterns, https://camunda.com/blog/2018/08/bpmn-for-microservices-orchestration-a-primer-part-1/ (last accessed: 2021-10-10), 2018.

[25] Camunda. Workflow and Decision Automation Platform, www.camunda.com, last accessed: 2021-10-10.

[26] Camunda. The Camunda Platform Manual Version 7.15 - Introduction, User Guide, Reference, https://docs.camunda.org/manual/7.15/, last accessed: 2021-10-10.

[27] Camunda. Camunda Optimize for Analyzing Business Processes, https://camunda.com/products/camunda-platform/optimize/, last accessed: 2021-10-10.

[28] A. Caracaş and A. Bernauer. Compiling Business Process Models for Sensor Networks. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8. IEEE, 2011.

[29] A. J. S. Cardoso. *Quality of Service and Semantic Composition of Workflows*. PhD thesis, University of Georgia, Athens, GA, USA, 2002.

[30] P. Cardoso, A. Respício, and D. Domingos. riskaBPMN - a BPMN Extension for Risk Assessment. *Procedia Computer Science*, 181:1247–1254, 2021.

[31] H. G. Ceballos, V. Flores-Solorio, and J. P. Garcia. A probabilistic BPMN normal form to model and advise Human Activities. In *International Workshop on Engineering Multi-Agent Systems*, pages 51–69. Springer, 2015.

[32] I. Chabini. Discrete Dynamic Shortest Path Problems in Transportation Applications: Complexity and Algorithms with Optimal Run Time. *Transportation Research Record*, 1645(1):170–175, 1998.

[33] N. Chakraborty, A. Mondal, and S. Mondal. Multiobjective Optimal Scheduling Framework for HVAC Devices in energy-efficient Buildings. *IEEE Systems Journal*, 13(4):4398–4409, 2019.

[34] C. Chang, S. N. Srirama, and R. Buyya. Mobile Cloud Business Process Management System for the Internet of Things: a Survey. *ACM Computing Surveys (CSUR)*, 49(4):1–42, 2016.

[35] S. Chhun, C. Cherifi, N. Moalla, and Y. Ouzrout. A Multi-Criteria Service Selection Algorithm for Business Process Requirements. *arXiv preprint arXiv:1505.03998*, 2015.

[36] H.-H. Chiu and M.-S. Wang. A Study of IoT-aware Business Process Modeling. *International Journal of Modeling and Optimization*, 3(3):238, 2013.

[37] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, et al. Web Services Description Language (WSDL) 1.1, 2001.

[38] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, et al. The Addition of Explicit Congestion Notification (ECN) to IP. *IETF RFC 3626*, 2003.

[39] J. C. N. Climaco and E. Q. V. Martins. A bicriterion Shortest Path Algorithm. *European Journal of Operational Research*, 11(4):399–404, 1982.

[40] Y. Cui, Z. Geng, Q. Zhu, and Y. Han. Multi-objective Optimization Methods and Application in Energy Saving. *Energy*, 125:681–704, 2017.

[41] F. Daniel, J. Eriksson, N. Finne, H. Fuchs, A. Gaglione, S. Karnouskos, P. M. Montero, L. Mottola, F. J. Oppermann, G. P. Picco, et al. makeSense: Real-world Business Processes through Wireless Sensor Networks. In *CONET/UBICITEC*, pages 58–72, 2013.

[42] D. Delling and D. Wagner. Pareto Paths with SHARC. In *International Symposium on Experimental Algorithms*, pages 125–136. Springer, 2009.

[43] D. Delling, T. Pajor, and R. F. Werneck. Round-based Public Transit Routing. *Transportation Science*, 49(3):591–604, 2015.

[44] R. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *International Conference on Business Process Management*, pages 48–63. Springer, 2009.

[45] R. M. Dijkman, M. Dumas, and C. Ouyang. Formal Semantics and automated Analysis of BPMN Process Models. *preprint*, 7115, 2007.

[46] E. A. Dinic. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. In *Soviet Math. Doklady*, volume 11, pages 1277–1280, 1970.

[47] Y. Disser, M. Müller-Hannemann, and M. Schnee. Multi-Criteria Shortest Paths in time-dependent Train Networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 347–361. Springer, 2008.

[48] Docker Inc. www.docker.com, last accessed: 2021-10-10.

[49] D. Domingos and F. Martins. Using BPMN to model Internet of Things Behavior within Business Process. *International Journal of Information Systems and Project Management*, 5(4):39–51, 2017.

[50] D. Domingos, A. Respício, and R. Martinho. Using Resource Reliability in BPMN Processes. *Procedia Computer Science*, 100:1280–1288, 2016.

[51] D. Domingos, A. Respício, F. Martins, and B. Melo. Automatic Decomposition of IoT Aware Business Processes - a Pattern Approach. *Procedia Computer Science*, 164:313–320, 2019.

[52] D. Domingos, A. Respício, and R. Martinho. Reliability of IoT-aware BPMN Healthcare Processes. In *Virtual and Mobile Healthcare: Breakthroughs in Research and Practice*, pages 793–821. IGI Global, 2020.

[53] J. Dörndorfer and C. Seel. A Meta Model based Extension of BPMN 2.0 for Mobile Context Sensitive Business Processes and Applications. *13. Internationale Tagung Wirtschaftsinformatik (WI2017)*, 2017.

[54] J. Dörndorfer and C. Seel. Context Modeling for the Adaption of Mobile Business Processes - An Empirical Usability Evaluation. *Information Systems Frontiers*, pages 1–16, 2020.

[55] J. Dörndorfer, C. Seel, and D. Hilpoltsteiner. SenSoMod - A Modeling Language for Context-Aware Mobile Applications. *Drews Paul, Funk Burkhardt, Niemeyer Peter und Xie Lin (Hg.): Multikonferenz Wirtschaftsinformatik (MKWI). Data Driven X-Turning Data into Value, Bd*, 4:6–9, 2018.

[56] J. Dörndorfer, F. Hopfensperger, and C. Seel. The SenSoMod-Modeler - A model-driven Architecture Approach for Mobile Context-Aware Business Applications. In *International Conference on Advanced Information Systems Engineering*, pages 75–86. Springer, 2019.

[57] Y. Dou, L. Zhu, and H. S. Wang. Solving the Fuzzy Shortest Path Problem using Multi-Criteria Decision Method based on Vague Similarity Measure. *Applied Soft Computing*, 12(6):1621–1631, 2012.

[58] O. Dousse, P. Thiran, and M. Hasler. Connectivity in Ad-Hoc and Hybrid Networks. In *Proceedings of the Twenty-first Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1079–1088. IEEE, 2002.

[59] S. E. Dreyfus. An Appraisal of some Shortest-Path Algorithms. *Operations Research*, 17(3):395–412, 1969.

[60] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management*. Springer, second edition, 2018.

[61] M. Dumas, M. Rosa, J. Mendling, and H. Reijers. *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, 2018. ISBN 9783662565094.

[62] H. Endert, T. Küster, B. Hirsch, and S. Albayrak. Mapping BPMN to Agents: An Analysis. *Agents, Web-Services, and Ontologies Integrated Methodologies*, pages 43–58, 2007.

[63] D. Eppstein. Finding the k Shortest Paths. *SIAM Journal on computing*, 28(2): 652–673, 1998.

[64] K. Erciyes. *Guide to Graph Algorithms: Sequential, Parallel and Distributed*. Texts in Computer Science. Springer International Publishing, 2019. ISBN 9783030103385.

[65] S. Even. *Graph Algorithms*. Cambridge University Press, 2011.

[66] K. Fall. A delay-tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 27–34. ACM, 2003.

[67] K. Fall, W. Hong, and S. Madden. Custody Transfer for Reliable Delivery in Delay Tolerant Networks. *IRB-TR-03-030, July*, 2003.

[68] K. Fall, K. Scott, S. Burleigh, L. Torgerson, V. Cerf, A. Hooke, H. Weiss, and R. Durst. Delay-Tolerant Networking Architecture. *IETF RFC 4838*, 2007.

[69] R. T. Fielding. REST APIs must be hypertext-driven, (last accessed: 2021-10-10), https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven/, 2008.

[70] L. R. Ford and D. R. Fulkerson. Maximal Flow through a Network. In *Classic Papers in Combinatorics*, pages 243–248. Springer, 2009.

[71] M. Fowler. CircuitBreaker, https://martinfowler.com/bliki/CircuitBreaker.html (last accessed: 2021-10-10), 2014.

[72] M. Fruhner, T. Iggena, F. Kraatz, F. Nordemann, H. Tapken, and R. Tönjes. OPeRAte: An IoT Approach towards Collaborative, Manufacturer-independent Farming 4.0. In *Proceedings of the 4^{th} International Conference on Internet of Things, Big Data and Security (IoTBDS)*, pages 165–176, 2019.

[73] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998.

[74] M. Gast. *802.11 Wireless Networks: The Definitive Guide*. A Nutshell Handbook. O'Reilly Media, 2005. ISBN 9780596100520.

[75] M. Geiger, S. Harrer, J. Lenhard, and G. Wirtz. On the Evolution of BPMN 2.0 Support and Implementation. In *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 101–110. IEEE, 2016.

[76] M. Geiger, S. Harrer, J. Lenhard, and G. Wirtz. BPMN 2.0: The State of Support and Implementation. *Future Generation Computer Systems*, 80:250–262, March 2018.

[77] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and simpler Hierarchical Routing in Road Networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[78] P. Ginzboorg, T. Kärkkäinen, A. Ruotsalainen, M. Andersson, and J. Ott. DTN Communication in a Mine. In *2nd Extreme Workshop on Communications*, 2010.

[79] A. V. Goldberg and C. Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, volume 5, pages 156–165. Citeseer, 2005.

[80] A. Gounaris. Towards Automated Performance Optimization of BPMN Business Processes. In *East European Conference on Advances in Databases and Information Systems*, pages 19–28. Springer, 2016.

[81] I. Graja, S. Kallel, N. Guermouche, and A. H. Kacem. BPMN4CPS: A BPMN Extension for Modeling Cyber-Physical Systems. In *25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 152–157. IEEE, 2016.

[82] I. Graja, S. Kallel, N. Guermouche, S. Cheikhrouhou, and A. Hadj Kacem. A comprehensive Survey on Modeling of Cyber-Physical Systems. *Concurrency and Computation: Practice and Experience*, 32(15):e4850, 2020.

[83] S. Grasic and A. Lindgren. Revisiting a Remote Village Scenario and its DTN Routing Objective. *Computer Communications*, 48:133–140, 2014.

[84] P. Hansen. Bicriterion Path Problems. In *Multiple Criteria Decision Making Theory and Application*, pages 109–127. Springer, 1980.

[85] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[86] S. Hinck, F. Nordemann, F. Kraatz, T. Iggena, R. Tönjes, H. Tapken, and D. Kümper. User-oriented, Web-based GIS Application for Liquid Manure Fertilisation. In *12th European Conference on Precision Agriculture (ECPA), Book of Posters*, pages 96–97, 2019.

[87] T. K. Hua and N. Abdullah. Weighted Sum-Dijkstra's Algorithm in Best Path Identification based on Multiple Criteria. *J. Comput. Sci. Comput. Math*, 8(3): 2–8, 2018.

[88] C.-M. Huang, K.-c. Lan, and C.-Z. Tsai. A Survey of Opportunistic Networks. In *22nd International Conference on Advanced Information Networking and Applications-Workshops (AINA Workshops 2008)*, pages 1672–1677. IEEE, 2008.

[89] P. Hui, J. Crowcroft, and E. Yoneki. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 241–250. ACM, 2008.

[90] P. Hui, A. Lindgren, and J. Crowcroft. Empirical Evaluation of Hybrid Opportunistic Networks. In *2009 First International Communication Systems and Networks and Workshops*, pages 1–10. IEEE, 2009.

[91] International Organization for Standardization (ISO). ISO:IEC 19510:2013, Information technology - Object Management Group Business Process Model and Notation. 2013.

[92] M. A. Islam. *Routing Issues in Opportunistic Networks: Evolution of Delay Tolerant Networks from Mobile Ad-hoc Networks to Opportunistic Networks*. LAP LAMBERT Academic Publishing, 2012.

[93] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. *SIGCOMM Comput. Commun. Rev.*, 34(4):145–158, Aug. 2004. ISSN 0146-4833.

[94] M. Jesús-Azabal, J. Berrocal-Olmeda, J. García-Alonso, and J. Galán-Jiménez. A Self-sustainable DTN Solution for Isolation Monitoring in Remote Areas. In *International Workshop on Gerontechnology*, pages 57–68. Springer, 2020.

[95] JGraphT-Library. Java library of graph theory data structures and algorithms, https://jgrapht.org, last accessed: 2021-10-10.

[96] R. Johari, P. Garg, R. Bhatia, K. Gupta, and A. Fatimah. Applications of DTN. In *Opportunistic Networks*, pages 260–266. Chapman and Hall/CRC, 2018.

[97] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, pages 153–181. Springer, 1996.

[98] G. Kougka and A. Gounaris. Optimization of Data Flow Execution in a Parallel Environment. *Distributed and Parallel Databases*, 37(3):385–410, 2019.

[99] G. Kougka, K. Varvoutas, A. Gounaris, G. Tsakalidis, and K. Vergidis. On Knowledge Transfer from cost-based Optimization of data-centric Workflows to Business Process Redesign. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLIII*, pages 62–85. Springer, 2020.

[100] E. Kuiper and S. Nadjm-Tehrani. Geographical Routing with Location Service in intermittently connected MANETs. *IEEE Transactions on Vehicular Technology*, 60(2):592–604, 2010.

[101] A. Lindgren, A. Doria, and O. Schelen. Probabilistic Routing in intermittently connected Networks. *Service Assurance with Partial and Intermittent Resources*, pages 239–254, 2004.

[102] L. Liu, H. Mu, X. Yang, R. He, and Y. Li. An Oriented Spanning Tree based Genetic Algorithm for Multi-Criteria Shortest Path Problems. *Applied Soft Computing*, 12(1):506–515, 2012.

[103] N. Lohmann, E. Verbeek, and R. Dijkman. Petri Net Transformations for Business Processes - A Survey. In *Transactions on Petri Nets and other Models of Concurrency II*, pages 46–63. Springer, 2009.

[104] R. Mangrulkar and M. Atique. Routing Protocol for Delay Tolerant Network: A Survey and Comparison. In *2010 International Conference on Communication Control and Computing Technologies*, pages 210–215. IEEE, 2010.

[105] M. Marchese, F. Patrone, and M. Cello. DTN-based Nanosatellite Architecture and Hot Spot Selection Algorithm for Remote Areas Connection. *IEEE Transactions on Vehicular Technology*, 67(1):689–702, 2017.

[106] R. Martinho and D. Domingos. Quality of Information and Access Cost of IoT Resources in BPMN Processes. *Procedia Technology*, 16:737–744, 2014.

[107] R. Martinho, D. Domingos, and J. Varajão. CF4BPMN: a BPMN Extension for Controlled Flexibility in Business Processes. *Procedia Computer Science*, 64: 1232–1239, 2015.

[108] R. Martinho, D. Domingos, and A. Respício. Evaluating the Reliability of Ambient-Assisted Living Business Processes. In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS)*, pages 528–536, 2016.

[109] E. Q. V. Martins. On a Multicriteria Shortest Path Problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

[110] F. Martins and D. Domingos. Modelling IoT Behaviour within BPMN Business Processes. *Procedia Computer Science*, 121:1014–1022, 2017.

[111] C. P. Mayer. *Hybrid Routing in Delay Tolerant Networks*. KIT Scientific Publishing, 2012.

[112] L. Mazzola, P. Kapahnke, P. Waibel, C. Hochreiner, and M. Klusch. FCE4BPMN: On-demand QoS-based optimised Process Model Execution in the Cloud. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 305–314. IEEE, 2017.

[113] S. Meyer, A. Ruppen, and C. Magerkurth. Internet of Things-aware Process Modeling: integrating IoT Devices as Business Process Resources. In *International Conference on Advanced Information Systems Engineering*, pages 84–98. Springer, 2013.

[114] S. Meyer, A. Ruppen, and L. Hilty. The Things of the Internet of Things in BPMN. In *International Conference on Advanced Information Systems Engineering*, pages 285–297. Springer, 2015.

[115] H. Mili, G. Tremblay, G. B. Jaoude, É. Lefebvre, L. Elabed, and G. E. Boussaidi. Business Process Modeling Languages: Sorting through the Alphabet Soup. *ACM Computing Surveys (CSUR)*, 43(1):1–56, 2010.

[116] T. Narendra, P. Agarwal, M. Gupta, and S. Dechu. Counterfactual Reasoning for Process Optimization using Structural Causal Models. In *International Conference on Business Process Management*, pages 91–106. Springer, 2019.

[117] S. Newman. *Building Microservices: designing fine-grained Systems*. O'Reilly Media, 2015.

[118] N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai. Adaptive Algorithms for detecting Community Structure in dynamic social Networks. In *2011 Proceedings IEEE INFOCOM*, pages 2282–2290. IEEE, 2011.

[119] F. Nordemann. Proof-of-concept implementation - an agricultural slurry process, https://github.com/fnordemann/ResilientProcessExecution.

[120] F. Nordemann, T. Iggena, F. Kraatz, M. Fruhner, H. Tapken, and R. Tönjes. Digitale Agrarprozesse für eine nachhaltige und verordnungskonforme Landwirtschaft am Beispiel einer kooperativen Flüssigmistausbringung. In *40. GIL-Jahrestagung, Gesellschaft für Informatik in der Land-, Forst-, und Ernährungswirtschaft*, 2020.

[121] F. Nordemann, R. Tönjes, and E. Pulvermüller. Resilient BPMN: Robust Process Modeling in Unreliable Communication Environments. In *8th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. Scitepress, 2020.

[122] OASIS. Web Services Business Process Execution Language (WS-BPEL) Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (last accessed: 2021-10-10), 2007.

[123] Object Management Group (OMG). Business Process Model and Notation (BPMN) 1.0 Specification, www.omg.org/spec/BPMN/1.0/About-BPMN (last accessed: 2021-10-10), 2006.

[124] Object Management Group (OMG). Business Process Model and Notation (BPMN) 2.0 Specification, www.omg.org/spec/BPMN/2.0/About-BPMN (last accessed: 2021-10-10), 2011.

[125] Object Management Group (OMG). Business Process Model and Notation (BPMN) 2.0.2 Specification, www.omg.org/spec/BPMN/2.0.2/About-BPMN (last accessed; 2021-10-10), 2014.

[126] Object Management Group (OMG). Case Management Model and Notation (CMMN) Spezification 1.1, www.omg.org/spec/CMMN (last accessed: 2021-10-10), 2016.

[127] Object Management Group (OMG). MetaObject Facility Specification 2.5.1, www.omg.org/mof (last accessed: 2021-10-10), 2016.

[128] Object Management Group (OMG). Unified Modeling Language (UML) 2.5.1 Specification, www.omg.org/spec/UML/About-UML (last accessed: 2021-10-10), 2017.

[129] Object Management Group (OMG). Decision Model and Notation (DMN) 1.3 Specification, www.omg.org/spec/DMN (last accessed: 2021-10-10), 2021.

[130] H. Ochiai, H. Ishizuka, Y. Kawakami, and H. Esaki. A DTN-based Sensor Data Gathering for Agricultural Applications. *IEEE Sensors Journal*, 11(11):2861–2868, 2011.

[131] OpenAPI Initiative. http://coreos.com/rkt/, last accessed: 2021-10-10.

[132] OPeRAte. Osnabrueck University of Applied Sciences: OPeRAte research project, http://operate.edvsz.hs-osnabrueck.de, last accessed: 2021-10-10.

[133] J. Ott, D. Kutscher, and C. Dwertmann. Integrating DTN and MANET Routing. In *Proceedings of the 2006 SIGCOMM Workshop on Challenged Networks*, pages 221–228, 2006.

[134] Pandas-Framework. Description of boxplots, https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html, last accessed: 2021-10-10.

[135] L. Pelusi, A. Passarella, and M. Conti. Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks. *IEEE Communications Magazine*, 44(11):134–141, 2006.

[136] C. Perkins, E. Belding-Royer, and S. Das. IETF Request for Comments (RFC) 3561: Ad Hoc On-demand Distance Vector (AODV) Routing, https://datatracker.ietf.org/doc/html/rfc3561 (last accessed: 2021-10-10), 2003.

[137] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, 1994.

[138] Pivotal Software. Spring Framework, http://spring.io, last accessed: 2021-10-10.

[139] S. D. Pohekar and M. Ramachandran. Application of Multi-Criteria Decision Making to Sustainable Energy Planning - A Review. *Renewable and Sustainable Energy Reviews*, 8(4):365–381, 2004.

[140] Postman, Inc. API Development Environment, https://www.postman.com, last accessed: 2021-10-10.

[141] S. Qin, Z. Zhao, J. Su, and Z. Yang. The Segmentation and Reconstruction Method of Business Process BPMN under Constraint Conditions. In *2021 International Conference on Service Science (ICSS)*, pages 45–50. IEEE, 2021.

[142] C. Raffelsberger and H. Hellwagner. A hybrid MANET-DTN Routing Scheme for Emergency Response Scenarios. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 505–510. IEEE, 2013.

[143] RAML Workgroup. http://raml.org, last accessed: 2021-10-10.

[144] Raspberry Pi Foundation. Raspberry Pi Zero W(H) Product Website and Specification, www.raspberrypi.com/products/raspberry-pi-zero-w/, last accessed: 2021-10-10.

[145] Red Hat Inc. jBPM - Open Source Business Automation Toolkit, www.jbpm.org, last accessed: 2021-10-10.

[146] Red Hat Inc. rkt, http://coreos.com/rkt/, last accessed: 2021-10-10.

[147] L. B. Reinhardt and D. Pisinger. Multi-Objective and Multi-Constrained non-additive Shortest Path Problems. *Computers & Operations Research*, 38(3):605–616, 2011.

[148] Request for Comments. RFC 791 Internet Protocol (IP), (last accessed: 2021-10-10) https://datatracker.ietf.org/doc/html/rfc791, 1981.

[149] Request for Comments. RFC 793 Transmission Control Protocol (TCP), (last accessed: 2021-10-10) https://datatracker.ietf.org/doc/html/rfc793, 1981.

[150] A. Respício and D. Domingos. Reliability of BPMN Business Processes. *Procedia Computer Science*, 64:643–650, 2015.

[151] S. Rubaiee and M. B. Yildirim. An energy-aware Multiobjective Ant Colony Algorithm to minimize total Completion Time and Energy Cost on a Single-Machine Preemptive Scheduling. *Computers & Industrial Engineering*, 127:240–252, 2019.

[152] I. B. Said, M. Chaabane, R. Bouaziz, and E. Andonoff. Flexibility of Collaborative Processes using Versions and Adaptation Patterns. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 400–411. IEEE, 2015.

[153] I. B. Said, M. A. Chaâbane, E. Andonoff, and R. Bouaziz. BPMN4V for Modeling and Handling Versions of BPMN Collaborations and Choreographies. In *International Conference on E-Business and Telecommunications*, pages 99–123. Springer, 2016.

[154] R. Sedgewick and K. Wayne. *Algorithms*. Addison-Wesley Professional, 4th edition, 2011. ISBN 032157351X, 9780321573513.

[155] N. Shi, S. Zhou, F. Wang, Y. Tao, and L. Liu. The Multi-Criteria Constrained Shortest Path Problem. *Transportation Research Part E: Logistics and Transportation Review*, 101:13–29, 2017.

[156] Signavio. Business Process Software, www.signavio.com, last accessed: 2021-10-10.

[157] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and Wait: an efficient Routing Scheme for intermittently connected Mobile Networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 252–259. ACM, 2005.

[158] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. A BPMN 2.0 Extension to define the Resource Perspective of Business Process Models. In *XIV Congreso Iberoamericano en Software Engineering*, 2011.

[159] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. Extending BPMN 2.0: Method and Tool Support. In *International Workshop on Business Process Modeling Notation*, pages 59–73. Springer, 2011.

[160] C. T. Sungur, P. Spiess, N. Oertel, and O. Kopp. Extending BPMN for Wireless Sensor Networks. In *2013 IEEE 15th Conference on Business Informatics*, pages 109–116. IEEE, 2013.

[161] A. Tovar, T. Friesen, K. Ferens, and B. McLeod. A DTN Wireless Sensor Network for Wildlife Habitat Monitoring. In *CCECE 2010*, pages 1–5. IEEE, 2010.

[162] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson, and B. Plattner. A Decade of Research in Opportunistic Networks: Challenges, Relevance, and Future Directions. *IEEE Communications Magazine*, 55(1):168–173, 2017.

[163] G. Tsakalidis, K. Vergidis, G. Kougka, and A. Gounaris. Eligibility of BPMN Models for Business Process Redesign. *Information*, 10(7):225, 2019.

[164] G. Tsakalidis, N. Nousias, and K. Vergidis. An inclusive Representation Approach to assess the Redesign Capacity of BPMN Models. 2020.

[165] A. Vahdat, D. Becker, et al. Epidemic Routing for partially connected Ad Hoc Networks. Technical report, Citeseer, 2000.

[166] W. Van Der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 2. Springer, 2011.

[167] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, et al. Process mining manifesto. In *International Conference on Business Process Management*, pages 169–194. Springer, 2011.

[168] W. M. Van der Aalst. Business Process Management: a comprehensive Survey. *ISRN Software Engineering*, 2013.

[169] A. Warburton. Approximation of Pareto Optima in Multiple-Objective Shortest-Path Problems. *Operations Research*, 35(1):70–79, 1987.

[170] Web Application Description Language (WADL). http://javaee.github.io/wadl/, last accessed: 2021-10-10.

[171] M. Weske. *Business Process Management*. Springer Berlin Heidelberg, 2019.

[172] Workflow Management Coalition (WfMC). Terminology & glossary. *WFMC Document WFMCTC-1011, Workflow Management Coalition, Avenue Marcel Thiry*, 204:1200, 1996.

[173] Workflow Management Coalition (WfMC). XML Process Definition Language (XPDL) 2.2 Specification, 2012.

[174] S. Yang. *Wireless Sensor Networks: Principles, Design and Applications*. Signals and Communication Technology. Springer London, 2013. ISBN 9781447155058.

[175] S. Yang, H. Cheng, and F. Wang. Genetic Algorithms with Immigrants and Memory Schemes for Dynamic Shortest Path Routing Problems in Mobile Ad Hoc Networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):52–63, 2009.

[176] A. Yousfi, A. De Freitas, A. K. Dey, and R. Saidi. The Use of Ubiquitous Computing for Business Process Improvement. *IEEE Transactions on Services Computing*, 9(4):621–632, 2015.

[177] A. Yousfi, C. Bauer, R. Saidi, and A. K. Dey. uBPMN: A BPMN Extension for Modeling Ubiquitous Business Processes. *Information and Software Technology*, 74:55–68, 2016.

[178] A. Yousfi, R. Saidi, and A. K. Dey. Variability Patterns for Business Processes in BPMN. *Information Systems and e-Business Management*, 14(3):443–467, 2016.

[179] A. Yousfi, M. Hewelt, C. Bauer, and M. Weske. Toward uBPMN-based Patterns for Modeling Ubiquitous Business Processes. *IEEE Transactions on Industrial Informatics*, 14(8):3358–3367, 2017.

[180] K. Zarour, D. Benmerzoug, N. Guermouche, and K. Drira. A Systematic Literature Review on BPMN Extensions. *Business Process Management Journal*, 2019.

[181] Z. Zhang. Routing in intermittently connected Mobile Ad Hoc Networks and Delay Tolerant Networks: Overview and Challenges. *IEEE Communications Surveys Tutorials*, 8(1):24–37, 2006.

[182] Y. Zhu, B. Xu, X. Shi, and Y. Wang. A Survey of social-based Routing in Delay Tolerant Networks: positive and negative Social Effects. *IEEE Communications Surveys and Tutorials*, 2012.

[183] M. Ziegelmann. *Constrained Shortest Paths and Related Problems: Constrained Network Optimization.* AV Akademikerverlag GmbH & Company KG, 2007. ISBN 9783836446334.

PUBLICATIONS

The core results of this thesis have been published in:

(1) Nordemann, Frank; Tönjes, Ralf; Pulvermüller, Elke: *Resilient BPMN: Robust Process Modeling in Unreliable Communication Environments.* In: Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development. Valletta, Malta, 02/2020, SCITEPRESS - Science and Technology Publications.

(2) Nordemann, Frank; Tönjes, Ralf; Pulvermüller, Elke; Tapken, Heiko: *A Graph-based Approach for Process Robustness in Unreliable Communication Environments.* In: Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering. Prague, Czech Republic, 05/2020, SCITEPRESS - Science and Technology Publications.

(3) Nordemann, Frank; Tönjes, Ralf; Pulvermüller, Elke; Tapken, Heiko: *Graph-based Multi-Criteria Optimization for Business Processes.* In: Proceedings of the 10th International Symposium on Business Modeling and Software Design (BMSD). Berlin, Germany, 07/2020, Lecture Notes in Business Information Processing, Springer.

(4) Nordemann, Frank; Tönjes, Ralf; Pulvermüller, Elke; Tapken, Heiko: *Resilient Business Process Modeling and Execution using BPMN and Microservices.* In: Model-Driven Engineering and Software Development. Communications in Computer and Information Science (CCIS), Springer International Publishing, 2021.

(5) Nordemann, Frank; Tönjes, Ralf; Pulvermüller, Elke; Tapken, Heiko: *Resilient Process Modeling and Execution Using Process Graphs.* In: Evaluation of Novel Approaches to Software Engineering. Communications in Computer and Information Science (CCIS), Volume 1375, Springer International Publishing, 2021.

The author of this thesis contributed to the scientific community by the publication of work related to this thesis:

(1) F. Nordemann, T. Iggena, F. Kraatz, M. Fruhner, H. Tapken, R. Tönjes: *Digitale Agrarprozesse für eine nachhaltige und verordnungskonforme Landwirtschaft am Beispiel einer kooperativen Flüssigmistausbringung*, 40. GIL-Jahrestagung, Gesellschaft für Informatik in der Land-, Forst-, und Ernährungswirtschaft, Weihenstephan, Germany, February 2020.

(2) M. Fruhner; T. Iggena; F. Kraatz; F. Nordemann; H. Tapken; R. Tönjes: *OPeRAte: An IoT Approach towards Collaborative, Manufacturer independent Farming 4.0.* In: Proceedings of the 4$^{\text{th}}$ International Conference on Internet of Things, Big Data and Security. Heraklion, Crete, Greece, 5/2/2019.

(3) F. Kraatz; H. Tapken; F. Nordemann; T. Iggena; M. Fruhner; R. Tönjes: *An Integrated Data Platform for Agricultural Data Analyses based on Agricultural ISOBUS and ISOXML.* In: Proceedings of the 4$^{\text{th}}$ International Conference on Internet of Things, Big Data and Security. Heraklion, Crete, Greece, 5/2/2019.

(4) F. Nordemann, T. Iggena, F. Kraatz, H. Tapken, R. Tönjes: *Modellierung, Ausführung und Steuerung von kooperativen Agrarprozessen mit BPMN und MQTT*, 38. GIL-Jahrestagung, Gesellschaft für Informatik in der Land-, Forst-, und Ernährungswirtschaft, Kiel, Germany, February 2018.

(5) F. Nordemann, F. Kraatz, H. Tapken, R. Tönjes: *Ein modulares Framework zur Modellierung, Konfiguration und Regelung von kooperativen Agrarprozessen*, 74. Internationale VDI-Tagung Land-Technik, Cologne, Germany, November 2016.

(6) F. Kraatz, F. Nordemann, R. Tönjes: *Protection and Privacy of Data in Cooperative Agricultural Processes - the Challenges of the Future*, International Conference on Agricultural Engineering (CIGR AGEng), Aarhus, Denmark, June 2016.

(7) F. Nordemann, R. Tönjes: *Robust Communication for Agricultural Process Management in Rural Areas*, 73. International Conference on Agricultural Engineering, Hannover, Germany, November 2015.

(8) F. Nordemann, T. Iggena, R. Tönjes: *Data-Driven Routing for Delay-Tolerant Networks*, 20. ITG Fachtagung Mobilkommunikation, Osnabrück, Germany, May 2015.

(9) F. Kraatz, F. Nordemann, R. Tönjes: *Anbindung von ISOBUS-Geräten an ein Online Precision Farming System*, 35. GIL-Jahrestagung, Gesellschaft für Informatik in der Land-, Forst-, und Ernährungswirtschaft, Geisenheim, Germany, February 2015.

(10) F. Nordemann, T. Iggena, R. Tönjes: *Leistungsbewertung von Routingprotokollkonfigurationen in verzögerungstoleranten Netzen*, 19. ITG Fachtagung Mobilkommunikation, Osnabrück, Germany, May 2014.

(11) F. Kraatz, F. Nordemann, R. Tönjes: *Unterbrechungstolerante Kommunikationsmechanismen für die dynamische Komposition von mobilen Precision Farming Anwendungen*, 18. ITG Fachtagung Mobilkommunikation, Osnabrück, Germany, May 2013.

(12) F. Nordemann: *A Communication-Optimizing Middleware for Efficient Wireless Communication in Rural Environments*, Proceedings of the 9$^{\text{th}}$ Doctoral Symposium of the 13th ACM/IFIP/USENIX International Middleware Conference, Montreal, Canada, December 2012.

(13) F. Nordemann, F. Kraatz, R. Tönjes: *Kooperierende mobile Ad-Hoc-Netzwerke (MANETs) und drahtlose Sensornetzwerke (WSNs) im Umfeld der Agrarwirtschaft*, Workshop Drahtlose Sensor-Aktor-Netzwerke, 22. Internationale Wissenschaftliche Konferenz Mittweida, Mittweida, Germany, October 2012.

(14) F. Nordemann, R. Tönjes: *Bewertung von Routing Protokollen für Ad-hoc Netze in landwirtschaftlichen Anwendungsszenarien*, 17. ITG Fachtagung Mobilkommunikation, Osnabrück, Germany, May 2012.

(15) F. Nordemann, R. Tönjes: *Transparent and Autonomous Store-Carry-Forward Communication in Delay Tolerant Networks (DTNs)*, IEEE International Conference on Computing, Networking and Communications (ICNC), Maui, Hawaii, USA, February 2012.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AODV	Ad-hoc On-demand Distance Vector
AppMap	Application Map
BFS	Breadth-First Search
BPD	Business Process Diagram
BPM	Business Process Management
BPMN	Business Process Model and Notation
BPMN+X	Business Process Model and Notation plus Extension
CDF	Cumulative Distribution Function
CELL	Cellular Positioning Service
CH	Contraction Hierarchies
Com-Paths	Combined-Paths (Analysis)
CPS	Cyber-Physical System
CSPF	Constrained Shortest-Path First
DAG	Directed Acyclic Graph
DFS	Depth-First Search
DMN	Decision Model and Notation
DSDV	Destination-Sequenced Distance-Vector
DSR	Dynamic Source Routing
DTN	Delay-Tolerant Network
EPC	Event-driven Process Chain
FCS	Frame Check Sequence
GW	Gateway
HATEOAS	Hypermedia as the Engine of Application State
H-Layer	Horizontal Layer

IoT	Internet of Things
KDE	Kernel Density Estimation
KPI	Key Performance Indicator
KSP	K-Shortest Paths
LAB	Laboratory Service
LPF	Longest-Path First
LOC	Local Positioning Service
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
Max-Step	Maximum-Step (Analysis)
MDA	Model-driven Architecture
MGMT	Management (Entity)
MOF	Meta Object Facility
NIRS	Near-Infrared Spectroscopy Sensor
OLSR	Optimized Link State Routing
OppNet	Opportunistic Network
OMG	Object Management Group
PLCP	Physical Layer Convergence Protocol
PML	Process Modeling Language
QoI	Quality of Information
QoS	Quality of Service
rBPMN	*Resilient BPMN*
REF	Ingredients Reference Service
REST	Representational State Transfer
RTK	Real-Time Kinematic
SL	Support Level
SLA	Service Level Agreement
SPF	Shortest-Path First
SR	Service Registry
VPN	Virtual Private Network
V-Layer	Vertical Layer
WPM	Weighted Product Model
WSM	Weighted Sum Model
WSN	Wireless Sensor Network
XMI	XML Metadata Interchange
XSD	XML Schema Definition

LIST OF SYMBOLS

Resilience calculation of a message flow
(based on connectivity characteristic edge weights)

BW	Bandwidth
BW_{avg}	Average bandwidth
BW_{min}	Minimum bandwidth
$F_{(P_f,y)}$	Function to integrate the effects of transfer failures
F_h	Frame header size
F_{pl}	Frame payload size
M_s	Message size
N_f	Number of frames
N_m	Number of messages
$P_{BW_{min}}$	Probability of BW_{min}
P_d	Required delivery probability
P_f	Packet transfer failure probability
T_{adv}	Advanced time (required for a message transfer)
T_b	Basic time (required for a message transfer)
T_d	Maximum allowed delivery delay time (for a message transfer)
T_f	Failure recovery time
T_i	Message flow interval

Resilience metrics

n	Number of weighted path edges
P_a	Average resilience probability of a path
P_b	Boolean resilience probability of a path
P_e	Resilience probability of a path edge
P_h	Highest resilience probability in a path
P_l	Resilience probability level of a path
P_m	Median resilience probability of a path
P_p	Resilience probability of a path
P_r	Probability range of a path
R_a	Average resilience of a path
R_d	Difference of a path
R_e	Resilience of a path edge
R_h	Highest resilience in a path
R_l	Resilience level of a path
R_m	Median resilience of a path
R_r	Range of a path
R_t	(Total) Resilience of a path

Multi-criteria metrics

x	Criterion placeholder (e.g., $A \Rightarrow$ Accuracy)
C_a^x	Average path weight
C_h^x	Highest path weight
C_l^x	Lowest path weight
C_m^x	Median path weight
C_r^x	Range of path weights
C_t^x	Total path weight
C_w^x	Edge weight
P_a^x	Average probability of path weights
P_b^x	Boolean probability of path
P_h^x	Highest probability of path weights
P_l^x	Lowest probability of path weights
P_m^x	Median probability of path weights
P_r^x	Probability range of path weights
P_p^x	Probability of path
P_w^x	Probability of edge weight

Combined-paths calculation

m	Number of path segments to be combined
P_{je}	Resilience probability of a joint path edge
R_{je}	Resilience of a joint path edge

APPENDIX

A

EXECUTING THE PROOF-OF-CONCEPT IMPLEMENTATIONS

The proof-of-concept implementations are available at Github. The following instructions illustrate how to execute and control the proof-of-concept implementations.

Since setting up an unreliable communication environment for evaluating the proof-of-concept is cumbersome, the code is designed to be executable on a single system. A neighbor-service for the slurry spreader can add and delete neighbors, resulting in an emulated unreliable communication environment. By interfacing a proactive routing protocol used in an unreliable network, the code may be used for execution in real-world environments.

The evaluation of this thesis includes two implementations. While the *master* branch includes the graph-based multi-criteria process analysis of slurry process *S3-Exe*, the branch *phd-wsm* includes the second slurry process featuring a MGMT participant and decision-making based on WSM.

Run the Proof-of-Concept Implementations

First of all, the repository needs to be cloned to the system supposed to execute the proof-of-concept.
Clone the master branch for process S3-Exe (graph-based decision-making):

```
git clone
       https://github.com/fnordemann/ResilientProcessExecution.git
```

or clone and checkout the phd-wsm branch for process S4-Exe (WSM-based decision-making):

```
git clone --single-branch --branch phd-wsm
       https://github.com/fnordemann/ResilientProcessExecution.git
```

The following three options exist to run the proof-of-concept implementations. Afterward, the execution can be controlled and manipulated by the methods presented in the following section.

Option 1: Run with Docker

Docker integration facilitates the execution of the proof-of-concept by setting up all dependencies. **Attention**: Docker commands may take a long time to execute - it is recommended to increase the available resources in the docker settings.
Build the BPMN Container:

```
       docker-compose build
```

Run the container and expose all necessary ports (see the Single system port mapping section). If no output is desired, add the option ***-d*** *to the statement.*

```
       docker-compose up
```

To shut down the system, run:

```
       docker-compose down
```

Option 2: Run precompiled .jar-files

Users may run precompiled Java archives, avoiding the need for source code compilations. Shell scripts are provided for UNIX environments. **Requirements**: Java Runtime Environment (JRE, v1.8+)
Automatically trigger execution by running:

```
       start_precompiled_services.sh
```

Option 3: Compile and run Source Code

Requirements: Java Development Kit (JDK, v1.8+), Maven 3.5.0+

Compilation of Source Code

Automatically trigger compilation by running:

```
start_compilation.sh
```

Running compiled Source Code

Automatically trigger execution by running:

```
start_freshlycompiled_services.sh
```

Controlling Scenario Execution

Different options to control and manipulate the slurry process execution exist.

REST-Calls in Postman

REST-calls can be used to start a slurry process, to add/delete neighbor nodes, and to inspect Eureka server instances. A collection of REST calls can be imported into the program *Postman* (https://www.postman.com):

```
postman-rest-helpercalls -> Postman_REST_Helpercalls.
    postman_collection.json
```

Controlling and inspecting BPMN Processes running in Camunda BPM

The slurry spreader and the MGMT participant (only in process S4-Exe) execute BPMN processes that can be started/monitored from Camunda tools (for *username/-password*, use *demo/demo*). Use Camunda Tasklist to start a slurry process and Camunda Cockpit to inspect running processes at:

```
- S3-Exe/S4-Exe: http://localhost:8035
- MGMT: http://localhost:8025
- MGMT(L) on S3: http://localhost:8036
```

Inspecting Eureka Server Instances

The Eureka servers are providing information about their current status, including registered services:

```
- Eureka-Cloud: http://localhost:8020
- Eureka-S: http://localhost:8030
- Eureka-NIRS: http://localhost:8040
- Eureka-LOC: http://localhost:8050
```

Process and Service Logs

Process and service operation can be examined from logs at *logs -> xyz-service.txt*

Single-system Port Mapping

Some services also provide information by accessing their interfaces. Besides, port information is helpful to adapt scenario execution. The port mapping for the scenario participants is as follows:

```
- Locally emulated Cloud environment:
        - Eureka-Cloud: 8020
                - PF: 8024 (only in S3-Exe)
                - LAB: 8023 (only in S3-Exe)
                - REF: 8026
                - REF2: 8026 (only in S4-Exe)
                - CELL: 8028
                - CELL2: 8029 (only in S4-Exe)
- S3-Exe/S4-Exe:
        - Eureka: 8030
        - Neighbor-Service: 8031
        - Camunda: 8035
        - PF(L): 8038 (only in S3-Exe)
        - REF(L): 8037
- NIRS:
        - Eureka-NIRS: 8040
        - NIRS: 8045
- LOC:
        - Eureka-LOC: 8050
        - LOC: 8055
```

Adaptation of Proof-of-Concept Implementations

- Users may find the source code commentaries helpful for understanding/adapting the proof-of-concept.
- Real-world deployments require to adapt the network configuration of services *xyz-service->src->main->resources->application.yml*
- Service discovery and decision-making is part of the slurry spreader implementation and may be used in other services.

Used Software

- Java
- Spring Framework
- JGraphT Java Library
- Camunda BPM
- Docker (for executing the proof-of-concept)