Transactional Support for ad-hoc Cooperations in Mobile Environments

Dissertation zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften (Dr. rer. nat.) im Fachbereich Mathematik und Informatik der Freien Universität Berlin

vorgelegt von

Dipl.-Inform. Katharina Hahn

Berlin 2010
Erstgutachter: Prof. Dr.-Ing. Heinz F. Schweppe, Freie Universität Berlin
Zweitgutachter: Prof. Dr. Christian Becker, Universität Mannheim

Datum der Disputation 12. Juli 2010
Abstract

The advent of wireless networking technologies in combination with the decreasing size of yet more powerful computing devices have led to the emergence of new applications to be deployed in mobile environments. Users equipped with mobile devices are able to spontaneously collaborate with each other in an ad-hoc manner. However, when deploying cooperative applications in mobile environments, one has to provide suitable means to cope with the characteristics of these environments. Due to the mobility of participants and the wireless networking technologies, mobile networks are more dynamic than fixed networks. As resources of mobile networks expose a temporary nature, failures of any kind are no longer the exceptional case. Suitable forward error-handling mechanisms which still allow for successful execution of an application in case of failure, as well as backward failure-recovery mechanisms, which avoid inconsistent system states, have to be integrated.

In this thesis, an integrated approach of transactional support of ad-hoc collaborations with service discovery in mobile environments is presented. The objective is to ensure reliable support while respecting the autonomy of mobile devices. Ad-hoc collaborations are implemented as service compositions, specified as workflows. We present a service discovery protocol for ad-hoc scenarios which exploits the mobility of nodes: It adapts to the current context of nodes and thereby ensures high availability of information and decreases the number of messages if possible. On the other hand, it enables discovery and usage of remote services which are not in the direct vicinity of nodes. This protocol builds the foundation for ad-hoc collaboration, as composition at runtime is only possible, if services may be discovered in the first place. On the other hand, it allows for forward failure-handling, as it enables finding of alternatives.

The core contribution of this thesis is an adaptive workflow management system: It explores transactional properties of services and employs semi-atomicity as the correctness criterion to allow for reliable yet autonomous coupling of services. Workflows are verified at runtime. If the verification fails, they are adapted during the execution to ensure correct termination in any case. Furthermore, the adaptive workflow management system enables autonomy to participating devices whenever possible, thus abandons from tight coupling of services to transaction phases. We prove that our approach produces optimal results regarding the autonomy of services.

We present analytical and experimental evaluation results which confirm the applica-
bility of our integrated approach: As opposed to existing pessimistic approaches which ensure correctness by tight coupling of components, it considerably increases the autonomy of services. In comparison to optimistic approaches, it allows for integration of diverse (i.e., non-compensatable) services by still ensuring correct execution in any case. In summary, our approach is a hybrid approach which ensures correctness in any case yet autonomous coupling of services whenever possible.
Acknowledgements

I would like to express my appreciation and gratitude for those from whom I have profited in so many different aspects over the past years.

First and foremost, I would like to thank my advisor, Prof. Heinz Schweppe, for giving me the opportunity to work on this thesis. Without the countless, valuable conversations and his continuous support and commitment to refining the contents and the presentation of this thesis, it would not look like it does today.

My special thanks go to Prof. Christian Becker, for revising the thesis and many precious words of advice. His enthusiasm for research continuously motivated me throughout the past: Thank you for the gap in the clouds!

I would like to express my gratitude to my peers at the ag-db and the diploma students for insightful technical discussions and their support. Furthermore, I want to thank my colleagues and friends who have enriched my life with contentual collaboration as well as sincere, entertaining and happy diversion: Fabian, Marc, Georg - I am grateful for open doors and ears and for the friendship we share; Christian - thank you for so many discussions over coffee and relevant insights: Of course, you are right after all. Additionally, I would like to extend thanks to Olli for the most wonderful and comforting latte. My special thanks are due to Manuel without whom my time at FU would have been less joyful - thank you for sharing all ups and all downs. Furthermore, I would like to thank Sinikka whose warmth constantly created a wonderful atmosphere in 165.

I owe my deepest gratitude to Kirsten: Thank you for saving my life, for making the sun shine bright and for being the most incredible friend one could possibly ask for - plan K rocks! Likewise, I am not able to adequately express my appreciation for Nicole: Thank you for all your patience and support in every way and every aggregate state, and for making life colorful. Thank you for being who you are and for taking me as I am - all inclusive.

Last but not least, I would like to express my gratitude to Annalene, Willi, and Tobi. Without your love, your encouragement and unconditional care for me on my way I would not be where I am today. Thank you for any emotional, motivational - but also technical - support and for continuously pointing out important and beautiful aspects in life.
Contents

List of Figures  xi
List of Abbreviations xiii

1 Introduction 1
  1.1 Motivation 1
  1.2 Application Scenario - MoP 3
  1.3 Objectives 5
  1.4 Outline 6

2 System Model 7
  2.1 Network Model 7
  2.2 Component Model 8
  2.3 Cooperation Model 8
  2.4 Failure Model 9
  2.5 Challenges 10

3 Related Work 11
  3.1 Existing Standards for Web Services 11
    3.1.1 Web Service Transaction Framework: WS-Tx 11
    3.1.2 Composing Web Services 12
  3.2 Related Research Areas 13
    3.2.1 Transaction Models 13
    3.2.2 Composition of Services 16
    3.2.3 Verification of Composite Services 20
  3.3 Summary 22

4 Discovering Mobile Services 23
  4.1 Existing Approaches 23
  4.2 Adaptive Group-based Service Discovery: aGSD 25
    4.2.1 Group-based Service Discovery 25
    4.2.2 Exploitation of Mobility 27
  4.3 Evaluation 31
## Contents

4.3.1 Evaluation Setup .............................................. 31
4.3.2 Evaluating the Dynamic Configuration ......................... 32
4.3.3 Evaluation of Remote Service Requests ........................ 35
4.4 Summary .......................................................... 36

5 Formalizing Transactional Cooperation of Services ................. 37
5.1 Transactional Services ........................................... 37
  5.1.1 Service Model ............................................... 37
  5.1.2 Transactional Properties of Services ......................... 38
5.2 Transactional Composition of Services .......................... 42
  5.2.1 Control Flow Patterns ....................................... 42
  5.2.2 Transactional Pattern ........................................ 43
  5.2.3 Transactional Properties of Patterns ......................... 44
  5.2.4 Workflow Elements .......................................... 45
  5.2.5 Dependencies ................................................. 46
5.3 Specifying Correctness: Semi-Atomicity .......................... 48

6 Flexible Workflows to Guarantee Correct Execution ............... 51
6.1 Views of the Workflow ........................................... 51
  6.1.1 Workflow as a Tree ........................................... 52
  6.1.2 Data Dependency Graph $G_\omega(V,E)$ ......................... 52
  6.1.3 ATS View .................................................... 53
6.2 Verification of a Workflow ....................................... 54
  6.2.1 Conflict Elements ............................................ 54
  6.2.2 Verification Criterion: SAP .................................. 59
  6.2.3 Verifying a Workflow $\omega$ ................................... 64
6.3 A-priori Adaptation of the Workflow ............................. 65
  6.3.1 Minimal Set of Coordinated Elements ......................... 65
  6.3.2 ATS-Invariant Adaptations ................................... 67
  6.3.3 Adaptation Algorithm ........................................ 68
  6.3.4 Correctness of the Algorithm ................................ 77
6.4 Integrating Service Discovery in Adaptive Workflow Management ... 80
6.5 Adaptation at Runtime ............................................ 81

7 Implementing Adaptive Workflow Management ....................... 83
7.1 Formal Requirements ............................................. 83
  7.1.1 Transactional Properties of Services ......................... 83
  7.1.2 Control Flow Patterns ....................................... 85
  7.1.3 Transactional Composition of Services ....................... 86
7.2 Architecture ....................................................... 87
# List of Figures

1.1 MoP activity flows for Berlin. ........................................ 4
1.2 MOP example scenario. .................................................. 4
2.1 Underlying system model. .............................................. 7
3.1 WS-Coordination framework. .......................................... 11
4.1 Hierarchical grouping of services. .................................. 25
4.2 Service provider $sp$, node $n$, and their mobility information. 29
4.3 Searching for remote services. ....................................... 30
4.4 Request hits and packet loss varying $c_L$. ......................... 33
4.5 Sent advertisements and request hits varying $v_{max}$. ....... 34
4.6 Packet loss and expenses varying $c_F$. ........................ 34
4.7 Distance to provider for discovery and usage of remote service. 35
4.8 Comparison of distance between client and target area vs. client and provider. 36
5.1 State-machine of a single service. .................................. 37
5.2 Running example with transactional properties. .................. 41
5.3 Properties of patterns of the running example. .................. 46
5.4 Correct (unsuccessful) termination of the running example. .... 49
6.1 System architecture of the adaptive workflow management system. 51
6.2 Tree view $T_\omega$ of the running example. ....................... 52
6.3 Data dependency graph $D_\omega(V,E)$ of the running example. 53
6.4 ATS view $ATS_\omega$ of the running example. ....................... 53
6.5 Rearranging transactional conflict $\{e_1,e_2\}_C$ (a) in sequence (b) and (c). 55
6.6 Directed transactional conflict $\{e_1,e_2\}_C$ (a), enclosed in $WP_{subTA}$ (b). 56
6.7 Transitive conflict $\{e_1,e_3\}_C$. ........................................ 57
6.8 Enclosed conflict element forms conflicts (a,b), is not part of conflict (c). 58
6.9 Examples of a $WP_{SEQ}$ which are not (a) [are (b)] correct. .... 60
6.10 Examples of a $WP_{AND}$ which are not (a) [are (b)] correct. .... 62
6.11 Verification of $T_\omega$. ............................................... 64
6.12 Constellation of $WP_{XOR}$ with uncertain properties $p_T(XOR) = (?, 0)$. 69


<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1</td>
<td>$d_{AWM}$ of $\omega'$ without data dependencies, varying $n$.</td>
<td>144</td>
</tr>
<tr>
<td>D.2</td>
<td>$d_{AWM}$ of $\omega''$ (left) and $\omega'''$ (right) varying $n$.</td>
<td>144</td>
</tr>
<tr>
<td>D.3</td>
<td>$d_{AWM}$ of $\omega''$ (left) and $\omega'''$ (right), compared to the analytical approximation.</td>
<td>146</td>
</tr>
<tr>
<td>D.4</td>
<td>$d_{AWM}$ of $\omega''$ varying $p_{RD}$ compared to the analytical approximations.</td>
<td>146</td>
</tr>
<tr>
<td>D.5</td>
<td>$d_{AWM}$ of $\omega'''$ varying $p_{RD}$, compared the analytical approximations.</td>
<td>147</td>
</tr>
<tr>
<td>D.6</td>
<td>$p_{SA}$ of an $WP_{AND}$ and $WP_{SEQ}$ pattern varying the number of elements $n$.</td>
<td>148</td>
</tr>
<tr>
<td>D.7</td>
<td>$d_{AWM}$ of the order-to-delivery process, varying $p_{RC}$ and $p_{RD}$.</td>
<td>150</td>
</tr>
<tr>
<td>D.8</td>
<td>$p_{SA}$ of the order-to-delivery process, varying $p_{RC}$ and $p_{S}$.</td>
<td>151</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2PC</td>
<td>Two Phase Commit</td>
</tr>
<tr>
<td>ACID</td>
<td>Atomicity, Consistency, Isolation, Durability</td>
</tr>
<tr>
<td>aGSD</td>
<td>adaptive Group-Based Service Discovery</td>
</tr>
<tr>
<td>ATM</td>
<td>Advanced Transaction Model</td>
</tr>
<tr>
<td>ATS</td>
<td>Accepted Termination States</td>
</tr>
<tr>
<td>AWM</td>
<td>Adaptive Workflow Management</td>
</tr>
<tr>
<td>BPM</td>
<td>Business Process Modeling</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Execution Language</td>
</tr>
<tr>
<td>BTP</td>
<td>Business Transaction Protocol</td>
</tr>
<tr>
<td>CAN</td>
<td>Content Addressable Network</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>DAO</td>
<td>Data Access Object</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>GPSR</td>
<td>Greedy Perimeter Stateless Routing</td>
</tr>
<tr>
<td>GSD</td>
<td>Group-based Service Discovery</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communication</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>LBS</td>
<td>Location Based Service</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad-hoc NETwork</td>
</tr>
<tr>
<td>MDBS</td>
<td>MultiDatabase Systems</td>
</tr>
<tr>
<td>MoP</td>
<td>Mobile Planet</td>
</tr>
<tr>
<td>OSGI</td>
<td>Open Service Gateway Initiative</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>rpo</td>
<td>Representational Partial Order</td>
</tr>
<tr>
<td>SAP</td>
<td>Semi-Atomicity Preserving</td>
</tr>
<tr>
<td>SLP</td>
<td>Service Location Protocol</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SSDP</td>
<td>Simple Service Discovery Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>THP</td>
<td>Tentative Hold Protocol</td>
</tr>
<tr>
<td>TIP</td>
<td>Transaction Internet Protocol</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Description, Discovery and Integration</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>WS</td>
<td>Web Services</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

The advent of wireless networking technologies in combination with the decreasing size of yet more powerful computing devices enables a wide range of new application fields to assist users in everyday situations. Application support is no longer restrained to designated and well-equipped working areas, such as office environments. One key to success of such applications, is the potential of spontaneously interconnecting devices anywhere to enable ad-hoc collaboration of users. However, mobile networks hold different characteristics than conventional fixed-wired networks: Due to the mobility of users and the wireless networking technologies, they are more dynamic which leads to less stability of communication links and more communication failures. On account of the inherent network dynamics, the execution context is not known previously to runtime and might differ from execution to execution. This also holds for the heterogeneity of nodes which can be integrated in collaborative wireless computing. Existing portable devices range from powerful laptops, over Personal Digital Assistants (PDA) and smartphones to special-target, pervasive computing devices [WHC+99].

Due to these characteristics of mobile networks, increased autonomy and flexibility of mobile computing come at the cost of decreased reliability. In this thesis, we therefore focus on how to support reliable collaboration. The stated network characteristics have to be taken into account to adapt suitable mechanisms to the current situation. Due to the temporary nature of resources of mobile networks, failures of any kind are no longer the exceptional case. In this thesis, we integrate suitable forward failure-handling mechanisms which still allow for successful execution in case of failure. If this is not possible, appropriate backward failure-handling to avoid inconsistent system states have to be employed. With respect to the autonomy of mobile participants, conventional mechanisms which rely on blocking of resources are not suitable in mobile environments. To enable autonomous execution of participants, we refrain from tight coupling of participants to strict execution phases (e.g., working-phase and commit-phase, as e.g., mobile transaction models, cf. Chapter 3). It is desirable to soften strict requirements such as assurance of availability of participating entities or blocking of resources. However, this also infers relaxed correctness guarantees, such as for example strict atomic out-
1.1 Motivation

come (either commit or abort) of all participants as known from distributed databases.\textsuperscript{1} Thereby, we trade strict correctness for autonomy of devices. By exploring the execution context, guarantees can be adaptively given to suit the circumstances at runtime.

A standard of implementing distributed cooperation of heterogeneous devices in fixed-wired networks is following the architectural paradigm of service oriented architectures (SOAs). These have proven their suitability especially in business environments. SOAs aim at loosely coupling of components independent of their underlying soft- and hardware platforms. Applications can be built by interconnecting possibly distributed and dynamically discovered components to so-called \emph{composed services}. They are specified using workflow languages, e.g. BPEL \cite{OAS07} or BPM\textsuperscript{2}, and their execution is controlled by workflow execution engines, such as jBPM or Apache ODE\textsuperscript{3}. For discovery of existing services, designated repositories, such as UDDI\textsuperscript{4}, are employed.\textsuperscript{5}

Existing standards and techniques, the leading one Web Services (WS), have proven to be powerful for fixed networks. However, not all of them provide flexible means to cope with the characteristics of mobile networks, especially ad-hoc cooperations i.e., the dynamics of participants, including their availability and mobility. Targeted assistance for different aspects, such as reliable and correct service composition in the presence of service discovery at runtime, remains challenging.

Consider for example, a mobile travel agent system, which we call MoP (Mobile Planet): It supports users in finding points of interest and booking activities according to their defined preferences. During the day, their mobile device searches for available services and, if confirmed by the user, it queries information, book single activities or packages of activities. Interaction patterns with services range from simple information providers to composed services in which all components demand several interaction steps. Examples of such services include booking tickets for cultural or recreational events (e.g., a museum or a concert), finding appropriate transport facilities (such as public transportation facilities or taxis) and offer information according to the user’s current location (i.e., location based services – LBS). As some of them involve monetary costs, parts of such workflows need transactional support regarding failure atomicity: No payment is to be performed for a service which has not completed, and vice versa.

In a fixed network scenario, an application designer is able to integrate designated service providers (and alternatives for them) in the workflow, which are then conducted at runtime. In mobile networks, designated service providers are hard to integrate in such a workflow, as their availability might not be given at runtime or their existence might not even be known at design time. Thus, suitable service providers have to be dynamically discovered and bound at runtime. Additionally, in order to provide

\textsuperscript{1}See Chapter 3 for further discussion of the mentioned concepts.
\textsuperscript{2}http://www.bpm.com/
\textsuperscript{3}http://www.jboss.com/products/jbpm/, http://ode.apache.org/
\textsuperscript{4}http://www.uddi.org/pubs/uddi_v3.htm
\textsuperscript{5}For an introduction to existing standards, the reader is referred to \cite{CLSF05, Pap07}.
transactional guarantees for certain parts of the workflow, the capabilities of the services available at runtime (such as blocking of underlying resources or compensatability) have to be dynamically explored.

In this thesis, transactional support regarding failure atomicity for ad-hoc cooperations implemented as composition of services in mobile networks is studied. The objectives are to analyze mobile workflows to be able to infer suitable guarantees for these environments. Approaches to dynamically discover services, and efficient algorithms to support these guarantees are introduced. These focus on how to ensure reliable collaboration by still respecting the autonomy of mobile participants. They are evaluated in a variety of settings to confirm their suitability for ad-hoc collaboration in mobile environments.

1.2 Application Scenario - MoP

We recurrently refer to the following application scenario of tourists exploring sites, e.g. in Berlin. As nowadays, most people carry mobile devices which offer a variety of network interfaces, we propose to make use of its functionality, especially its networking capabilities. Our application scenario is the mobile planet (MoP), a mobile, electronic, and more sophisticated version of the travel guide Lonely Planet, acts as a mobile travel guide\(^6\). As it inheres additional knowledge about the user’s preferences and its current location, as well as the ability to gain contextual knowledge about the currently visiting site, it is able to assist the user in novel ways.

Services which are desirable to integrate in MoP are manifold. They range from simple information services, which provide information about e.g., weather conditions or entry fees of museums or sites such as the Siegessäule, to LBS, e.g., employing directions or maps and information about interesting sites nearby. On the other hand, a variety of business cases is desirable as well: Mobile ticket vendors are able to sell tickets for museums or theater shows, transportation (such as public transportation, i.e. BVG\(^7\) or taxi companies). Additionally, services which interconnect tourists, and search for e.g., companions, are supposable within the system.

It is of interest for mobile ticket vendors to promote their partner sites and sell tickets or packages of these. If a ticket is purchased, it is either printed or a digital entry code (e.g., such as an identification number, string, or a 2D barcode) is issued, which the user employs as its entry code. As monetary costs are involved, it is important to integrate failure-tolerance in order to achieve a reliable and viable system.

According to the user’s preferences and the present service providers, several activity flows\(^8\) of touristic living are suggested for the day (see Figure 1.1). At first, the system provides information about Berlin and the current surrounding of the user, including a...

\(^6\)©Lonely Planet Publications edited by MAIRUMONT GmbH & Co. KG
\(^7\)Berliner Verkehrsbetriebe
\(^8\)This term is used interchangeably with the term workflow.
map and points of interest. As it can be seen in Figure 1.1, MoP then proposes several plans for the day. If the user decides for a proposal, the included services are invoked to book tickets (e.g., Memorial Berlin-Hohenschönhausen, Berlin Underworld Association\textsuperscript{9}, Filmmuseum). Accordingly, transportation facilities are organized, tickets or tables at restaurants (e.g., Käfer, Bayrisches Haus) are reserved and the user is provided with information about the sites to visit.

Although mobile devices are likely to be equipped with GSM\textsuperscript{10} or UMTS\textsuperscript{11}, ad-hoc communication is to be favorably used in this scenario, to avoid possibly expensive roaming costs and to remain flexible in terms of service providers. Thus, as long as the value of the transaction does not exceed the roaming costs, ad-hoc communication is beneficial within delay tolerant applications.

Throughout this thesis, we recurrently refer to the following simple example scenario as depicted in Figure 1.2 to outline details of our approach. In this example, a mobile ticket vendor sells tickets to the Philharmonics. At the beginning, the customer’s requests are specified (CRS). In order for his offer to be more attractive, the vendor additionally offers to organize Transportation to the concert venue, and according to the customer’s preferences, reserves a table at a nearby restaurant (Reservation). The offer is then confirmed and delivered, i.e., tickets are either printed or transfered as eTickets

\textsuperscript{9}Berliner Unterwelten, www.berliner-unterwelten.de
\textsuperscript{10}Globals System for Mobile Communication
\textsuperscript{11}Universal Mobile Telecommunications System
1.3 Objectives

In this thesis, ad-hoc collaboration of devices in mobile environments is studied. Due to the inherent characteristics of mobile networks, it is desirable to identify and provide suitable transactional guarantees for participating entities. We thus aim at the following objectives:

**Ad-hoc Collaboration of Mobile Devices** The primary objective of this thesis is to identify and implement failure-tolerant support mechanisms for ad-hoc collaboration of mobile devices. The goal of collaboration assistance is to provide suitable transactional guarantees concerning failure atomicity for collaborating entities while still respecting their autonomy.

In order to reach such a viable collaboration assistance system, the following sub-ordinate objectives have to be considered.

**Reliable Collaboration** As mobile networks are more dynamic and less reliable than fixed-wired networks, *failure-tolerance* is one key concern when supporting mobile collaboration. Appropriate forward-failure handling mechanisms which still allow for successful interaction in the presence of failures are to be identified and integrated. Likewise in the presence of failures, it has to be identified which measures have to be taken to avoid *inconsistent* system states. This infers the specification of *correct* cooperation: I.e., when is a collaboration considered to be correctly finished (either successfully *completed* or *aborted*)?

**Respecting Autonomy** This notion of correctness is related to transactional guarantees, e.g. the ACID guarantees (Atomicity, Consistency, Isolation and Durability) demanded in (distributed) databases. However, protocols currently used to provide these guarantees are very strict concerning the requirements of participants, such as availability and their tight coupling to transactions. It is therefore desirable to identify means to ensure correctness by still respecting the autonomy of mobile participants, thus allowing for loosely coupling of components.

**Dynamic Discovery and Binding** In fixed-wired networks, participants of collaborations are most likely to be known prior to execution. In mobile environments, not all participants can be previously determined, thus appropriate mechanisms to discover and bind participants at runtime have to be developed. If more than one provider for a service is discovered, it is desirable to integrate several of them as alternatives, if they fulfill the demands of the cooperation in the current context.
Exploration of Participants As the execution context might differ from execution to execution, it is desirable to adapt cooperation assistance to the current context. To respect the autonomy of mobile devices, rather restrictive requirements, such as the capability of blocking resources or the assurance of availability for certain time periods are to be avoided – if possible. We therefore explore participants present and their non-functional properties which relate to the specified correctness criteria (e.g. compensatability) at runtime. We also aim at providing dynamic and flexible support at runtime, as participants (respectively, their properties) might differ from execution to execution.

1.4 Outline

The remainder of this thesis is structured as follows: In Chapter 2, we present our system model and introduce relevant terms and concepts. In Chapter 3, related work is outlined, including existing standards as well as related research areas. In Chapter 4, we concisely present our approach to discover services in mobile ad-hoc environments. The focus of this thesis lies on Chapter 5 and 6: Chapter 5 contains the formal model of transactional cooperation. Our approach to flexibly guarantee reliable service composition by respecting the autonomy of participants is presented in Chapter 6. The implementation realizing our approach is introduced in Chapter 7. A thorough evaluation comparing our algorithms to existing techniques is presented in Chapter 8. We conclude in Chapter 9.
2 System Model

The basic concepts of our system model are depicted in Figure 2.1. Ad-hoc *Cooperations* consist of collaborating entities also referred to as *components*. Components communicate with each other via wireless *network* channels.

![Figure 2.1: Underlying system model.](image)

2.1 Network Model

Participants of mobile networks, communicate either directly with each other or via base stations. The former case is also referred to as mobile ad-hoc networks (MANETs). Mobile networks differ from conventional networks mainly in their spontaneous and temporary nature. They only exist, as long as participants linger in communication ranges of others. In MANETs, nodes are able to directly communicate with others in their broadcast range (single-hop). If corresponding network functionality is provided, they may also communicate via several hops (multi-hop) [RP99, Joh94].

Mobile nodes are equipped with one or several wireless networking interfaces, e.g. WLAN or GSM, which all feature quite different characteristics in terms of bandwidth, communication range and costs. Eventually, some participants may be reachable via more than one channel at a time. The obstacles of mobile networks are the mobility of devices which leads to momentary (un-)availability of communication links and remote resources of any kind. As mobile networks are temporary in their existence, the execution environment (also referred to as the execution context) is unknown prior to execution.

We target applications which allow for delay tolerant networking. It is therefore favorable to use ad-hoc communication, if the costs of communicating (which explosively
increase, if roaming costs are involved) exceed the value of the application. By value, the monetary costs are denoted as well as personal benefits, which can only be estimated rather than actually billed. However, if ad-hoc communication continuously fails, we assume nodes to hark back to reliable network channels (such as GSM or UMTS) which are costly however dependable.

Please note, we motivate our work with applications which integrate mobile participants. Thus, we assume the execution environment to be dynamic in terms of participants. However, mobility of nodes is not a necessary precondition for our system. Using our formal model (see Chapter 5), especially the defined properties of services, we abstract from the underlying network and mobility of nodes. Therefore, our algorithms to support flexible yet reliable cooperation is also appropriate for applications in fixed-wired settings which desire flexibility at runtime.

2.2 Component Model

SOAs aim at loosely coupling of components. Especially in the presence of failures, dynamic and loosely binding of components increases flexibility and facilitates failure-tolerance. We therefore argue that SOAs are a promising approach to implement cooperation in mobile environments as they respect the autonomy of components.

Each ad-hoc collaboration consists of several participants, which we refer to as components or entities. These entities are considered to be implemented as services which are possibly running on different devices. Their underlying soft- and hardware as well as their operational mode is hidden behind a well-defined interface. We assume each service to provide functionality to be invoked (e.g., by the cooperation or other services) and returning of results. Further service features include e.g., certain failure handling capabilities such as compensatability.

Components of the system can either be services or composition of services thus encapsulating cooperations themselves. The key issues of this thesis is the coordination of components while respecting their autonomy. I.e., we refrain from tight coupling of components to the execution of other components (or coordinators) rather than enabling loosely coupling as originally intended by SOAs. Note, that components do not necessarily employ designated resource managers as it is the case e.g., for distributed databases. Components offer services published via their interface, however their backend systems and the protocols they employ, are invisible to others. A detailed definition of services, including their behavior and properties, is given in Chapter 5.

2.3 Cooperation Model

Ad-hoc cooperations between devices are implemented in applications which are (partly) built on distributed services. Thus, the ad-hoc cooperation as such is defined as a com-
posite service. The standard for specifying composite services are workflow languages e.g. BPEL. We assume participants of ad-hoc cooperations to be discovered at runtime. I.e., as opposed to conventional composite services, components are not statically integrated at design time rather than dynamically bound at runtime. Please note, the term ad-hoc cooperation does not infer, that entities communicate via ad-hoc networks. As stated above, we assume them to preferably use ad-hoc communication if the costs of the application do not exceed the costs for communicating over costly network channels.

By defining accepted termination states (ATS), requirements for successful execution of the cooperation are given. Thus, components which are imperative for completion, as well as alternatives and prioritization among them are specified. The workflow additionally defines control-flow dependencies and the data dependencies among its components. Through the syntax of the composition, the semantics (e.g., inclusion of all components as opposed to choosing alternatives) of the collaboration is implicitly given. Once, the collaboration has been initiated, the composite service is active and the ultimate goal is to successfully complete (also referred to as commit), i.e. reaching ATS.

As we target mobile environments, we assume components to be dynamic, i.e., they might not be known at design time (of the cooperation) or temporary unavailable at runtime. Cooperations have to be flexible enough to be dynamically composed at runtime and on the other hand appropriately react to failures. Failure-handling mechanism (recovery) are employed to avoid incorrect (or inconsistent) system states: Forward-recovery still allows for successful completion, alternatively backward-recovery resets the system to a previously consistent state. This is also considered to be correct termination of the workflow as inconsistencies are avoided. Typical recovery mechanisms are retrying the defective action, executing an alternative, cancel certain activities or compensating for an activity. Definitions of these terms is given in Chapter 5.

As stated before, components are implemented as services and cooperations assembled by composing services. A cooperation of participants is implemented as a workflow and its execution is carried out by a management system. We refer to such a system, which enables correct execution of workflows by utilizing the appropriate recovery mechanisms in the presence of failures as a transactional workflow management system.

2.4 Failure Model

Regarding the execution of an ad-hoc cooperation, the following failures may occur: Communication failures refer to situations, in which messages between entities are lost. Especially in mobile ad-hoc networks this may commonly occur. We assume these kind of failures to be identifiable. They are recovered by re-transmitting according messages. Nodes, which dispose reliable network channels, such as UMTS, hark back on these, if ad-hoc communication continuously fails.

Interaction with services may result in component failures, either if appropriate services
cannot be discovered or the invocation of them results in failure. Service failures occur either due to internal errors (e.g., tickets are no longer available) or externally triggered, e.g., the execution is intermitted by another service or a human. How these failure are dealt with depends on the service, the collaboration and the current execution context. This is subject to the algorithms introduced in Chapter 6.

If an ad-hoc collaboration is executed, it may overall fail due to communication or service failures (e.g., necessary services cannot be discovered). However, if a cooperation is not successfully finished, it may still correct if its state obeys the correctness criterion (semi-atomicity) specified in Chapter 5. A cooperation failure refers to the status of a finished cooperation which does not fulfill the correctness criterion. We guarantee correct execution of ad-hoc collaborations, thus avoid these failure by appropriate algorithms as introduced in Chapter 6.

2.5 Challenges

Considering the objectives (Section 1.3) in our system model, these challenges arise.

**Mobile Service Discovery** In order to cooperate with nodes currently present, one has to be able to discover available services at runtime. As mobile environments are dynamic in their existence, it is desirable to explore (and exploit) the current execution context in order to efficiently adapt ones strategies.

**Dynamic Transactional Support** Heterogeneity of mobile environments also comprises the diversity of requirements of present components. In order to transactionally support ad-hoc cooperations employing SOAs, it is indispensable to classify transactional demands of components and cooperations to adapt to the current execution context. Furthermore, it is indispensable to be able to cope with the dynamics of participants, i.e., unavailability of services and discovery of alternatives at runtime.

**Suitable Correctness Guarantees** A formal model and the contrasting comparison of the transactional classification of all participants are necessary to derive suitable correctness guarantees. These have to meet the requirements of dynamically bound services, as well as those of the workflow.

**Autonomy vs. Correctness** In all respects the targeted environments demand careful appreciation of correctness on the one hand and autonomy of devices on the other. This trade-off has to be carefully considered as neither incorrect nor too autonomy-limiting applications are viable in mobile environments. Proper verification means are required to validate workflows in their current execution context. Adaptation mechanisms have to be provided in order to flexibly alter the workflow to ensure correctness in case the verification fails.
3 Related Work

In this chapter, we present existing standards for Web Services, which are utilized to ensure transactional support for composite services. Furthermore, we relate our work to research areas and approaches which are closely related to our approach of transactional support of ad-hoc cooperations.

3.1 Existing Standards for Web Services

3.1.1 Web Service Transaction Framework: WS-Tx

Applications may be built on several interconnected WS which are likely to be provided by several nodes. In order to obtain consistent outcome, all involved services must universally agree on the decision. The WS-Transaction Framework (WS-Tx) comprises several standards in order to support different transactional demands. The relationship between these specifications is shown in Figure 3.1.

![Figure 3.1: WS-Coordination framework.](image)

The WS-Coordination Framework describes an extensible framework for providing protocols that coordinate actions of distributed applications. It employs two services, the Activation and Registration service, and an extensible set of Coordination Protocols to reach consistent agreement on the outcome of distributed activities.
3.1 Existing Standards for Web Services

The ActivationService is used by the application to create a new coordinated activity which contains important information about the activity. Other applications register for the activity using the RegistrationService. The coordination protocols define two different protocol families, which provide support for short-lived transactions (WS-AT) and long-lived business activities (WS-BA).

3.1.1.1 WS-AtomicTransaction

WS-AtomicTransaction (WS-AT) provides atomic transaction support for short-lived transactions. The specification defines different coordination protocols for participants: Completion is used by the application to complete an activity. Atomic commit of participants is ensured by using two slightly different variants of the two-phase commit protocol (2PC), namely volatile and durable 2PC. The variants specify the order in which participants are notified by the coordinator.

As WS-AT relies on 2PC, it implies that all participants block their resources. This approach is not suitable for ad-hoc cooperations in mobile networks, as blocking of resources limits the autonomy of participants.

3.1.1.2 WS-BusinessActivity

The WS-BusinessActivity (WS-BA) specification provides support for long-living distributed transactions which demand consistent outcome. This is achieved on the premise of doing work, so that it can be undone later. Thus, participants are requested to provide compensation functionality for completed operations. The specification defines two protocols: BusinessAgreementWithParticipantCompletion and BusinessAgreementWithCoordinatorCompletion. These protocols differ in terms of participation contracts: Within the participant completion protocol a participant may unilaterally decide to end its participation whereas that is not allowed when using the coordinator completion protocol.

WS-BA is designed for long-lived activities trying to remedy the drawback of blocking resources as done by WS-AT. However, it obliges that participants have to be able to compensate the completed work at a later point in time. Therefore, only participants fulfilling this assumption are able to be integrated in a coordinated business activity. Otherwise, one risks inconsistent system states as it is shown in Chapter 8.

3.1.2 Composing Web Services

By employing Web Services, applications may be built by composing several WS into a new value-added WS. The specification of WS distinguishes between two composition standards, namely orchestration and choreography. Choreography defines the message exchange between services, but it does not validate the process itself as a whole. It is therefore not suitable for transactional support of service composition. Orchestration
of services on the other hand emphasizes the coordination between different involved entities which is also used to support transactional composition of services.

Nowadays, the business process execution language (WS-BPEL) \cite{OAS07} is the standard for orchestration of WS. It provides the ability to define compositions as processes by integrating services in form of partnerLinks. It also enables, recursive definition of processes, specifies lifecycle management and recoverability of processes.

BPEL is an XML-based language, which is executed by workflow engines, such as jBPM or Apache ODE. The basic concepts of BPEL embrace activities and scopes. Activities are either basic activities (e.g., assigning values to variables) or structured activities, such as parallel or sequential alignment of operations. Scopes are employed to group activities: Thereby, certain behavior may be fine-granularly specified.

Additionally, BPEL supports event-based communication employing event handlers, fault handlers and compensation handlers. Event handlers realize time-based and message based asynchronous communication, while fault handlers are automatically invoked if faulty behavior occurs. Compensation handlers are responsible for reversing completed work of the current scope. Fault-handling is done hierarchically: If no fault handler is specified within the current scope, it is passed on to its parent scope.

The concept of handlers may be used by the designer to fine-granularly specify how to cope with faulty behavior, thereby enabling custom-built exception and failure handling. However, it does not provide means to verify existing compositions, provide transactional guarantees or flexibly react to changes at runtime. We argue, that failure handling can be automated by exploring services at runtime, and therefore provide suitable abstraction for failure handling and transactional guarantees which should not be the designers responsibility.

### 3.2 Related Research Areas

We relate our work to the following research areas. At first, we present transaction models for different environments. Furthermore, we characterize approaches published in the area of service composition. Along with these, many formalism have been introduced to enable verification of composite services to which we additionally relate our approach.

#### 3.2.1 Transaction Models

Traditional database transactions are typically flat in terms of participants and ensure the ACID properties. Advanced transactions models (ATM) extend and relax the classical database transactions in the following way. They are extended among others in terms of participants: In many of these models, means of grouping participants or hierarchical structures of subtransactions are provided. On the other hand, most advanced transaction models relax some of the ACID properties, ensuring e.g., relaxed...
atomicity or isolation. **Transactions in multidatabase systems** additionally cope with the heterogeneity of participants. Thus, they cover diverse requirements of subtransactions. **Mobile transaction models** target applications in mobile environment, i.e., mobile databases or mobile participants. They additionally integrate appropriate recovery mechanisms to be able to cope with unsteady availability and disconnection of participants.

### 3.2.1.1 Advanced Transaction Models

A vast amount of research papers in the late 1980s and early 1990s addressed advanced transaction models. We briefly characterize them and relate these to our work.¹

In *nested transaction* [Mos], subtransactions are hierarchically arranged. Subtransactions are started by their according parent transaction. In case of failure of a subtransaction, all of its child subtransactions are aborted, however the parent of a failing subtransaction may decide on which recovery measures to be taken. *Open nested transactions* as proposed by Weikum and Schek [WS92] further relax isolation by allowing the changes of committed subtransactions to be visible to top-level transactions. Thereby, the degree of allowed parallelism is increased.

The *SAGAS* transaction model, as introduced by Garcia-Molina [GMS87], splits a longrunning transaction into a sequence of subtransactions. Each of these is related to an according compensating subtransaction. In case of failure of a subtransaction, *semantic atomicity* [GM83] is ensured by executing all compensating subtransactions of previously committed transactions. The *ConTract* model [WR92] enhances SAGAS by control structures. They consist of *steps* whose execution is arranged in a *script*. Such a script has to be forward-recoverable, i.e., in case of failure, the script may be executed again from the point of failure. Similar, the *split-join* transaction model [PKH88] enables a transaction to split itself into two independent (or dependent) subtransactions which may be joined at a later point in time.

Reasoning about various transaction models is enabled by the *ACTA* framework [CR90]. It is a meta-model which characterizes important aspects of transactional support. Using a set of dependencies, the structure and the behavior of transactions are specified. ACTA can be used to decide whether a particular execution history obeys specified dependencies.

In [AAA⁺96], Alonso et al. discuss the use of ATMs in workflow contexts. They argue, that ATMs are only partially suitable for transactional support of workflows: Workflow management systems automate flow of control and data between activities. Additionally, they map activities to users and programs while existing ATMs limit themselves to well-defined failure semantics in the sense of concurrency control and recovery features. The authors state that all ATMs may be used in the context of workflow management,

¹For a general introduction to ATMs, we refer to [JK97, Elm92].
3.2 Related Research Areas

however do not fully solve the problem of transactional support. That is due to workflow management systems offering a much more comprehensive solution to application support than advanced transaction models. Gaaloul et al. [GRGH07] identify ATMs as too inflexible in terms that they are not able to incorporate different behavioral patterns as well as transactional semantics into a single transaction.

Besides these reasons, we feel that the unsuitability of ATMs for transactional support of ad-hoc cooperations mainly relies in our targeted system model: On the one hand, ATMs are designed to be deployed in tightly-coupled environments. That is, subtransactions are not autonomous entities rather than operations which are fully controlled by the coordinator. On the other hand, ATMs are designed for static situations: They are not capable of supporting ad-hoc cooperations, i.e. they are not flexible enough to integrate dynamic participants and accommodate their requirements at runtime.

3.2.1.2 Transactions in Multidatabase Systems

Transactions in multidatabase systems are designed to cope with heterogeneous environments. A multidatabase transaction consists of a global transaction controlled by a global entity (multidatabase system, MDBS) and several local transactions, controlled by local database systems [BGMS92]. Each local system operates autonomously without the knowledge of other local systems. Successful execution of a multidatabase transaction is represented by the commitment of a representational set of subtransactions. If all of them commit, the transaction is said to reach an accepted termination state.\(^2\)

Typically, all transaction models for multidatabase systems, e.g., [ZNBB94, ZNB01, EJK+96, MRKS92a, DU96] distinguish transactional properties of (among others compensatability) and certain relations between (e.g., preference, precedence) subtransactions. The semantics of the properties and relations are mostly similar, however their notation and names differs and the cited approaches. For a detailed survey of multidatabase transaction management, including challenges and approaches, we refer to [BGMS92].

In flexible transactions [ZNBB94, ZNB01], subtransactions are classified to be retrievable, compensatable or pivot. Among subtransactions preference and precedence relations are defined. According to these, sets of subtransaction whose execution reflects the successful execution of the global transaction are implicitly defined. The flexible transaction is considered to be semi-atomically completed, if either no subtransaction is completed, or all subtransactions in one such set are committed and no other subtransaction is completed. In the model of flexible transactions, semi-atomicity is validated by reviewing the order of subtransactions according to their properties: The commitment of compensatable subtransactions precedes the commitment of pivot subtransactions. As their commitment infers the completion of the whole transaction, it is only followed

\(^2\)In the literature, it sometimes also referred to as committed acceptable termination state.
by retrievable subtransactions.

Flexible transactions are already close to our requirements. However, as with the other models for multidatabase transactions, they focus on verification of firm structures and global serialization and deadlock detection respectively. Their system model has in common with that of ATMs that it assumes statically defined structures of the transaction. Dynamic binding of participants and thus flexibility at runtime cannot be integrated.

### 3.2.1.3 Mobile Transaction Models

There exist a lot of mobile commit-protocols, which concentrate on reaching atomic outcome in environments with fluctuant availability of participants, such as e.g. [PA02, BGO07] or how to build reliable recommendation systems in order to enable fair transactions, e.g., [HKRBO07, AGG+05, AGG+04]. For a comparison of transaction commit protocols in mobile environments we refer to [BLRSA04].

Additionally to these protocols, there exists a vast amount of transaction models for mobile environments, such as CheeTah [PA00], MoFlex [KK00b], Team [GGGG04] and Kangaroo Transactions [DHB97] and many others, e.g., [GB01, Muk02]. For an introduction to these transaction models, we refer to [MS04, HTKR05]. A detailed survey which compares some of these models may be found in [SARA04].

These have in common, that they schedule activities according to resource constraints of devices and further relax isolation requirements in order to still guarantee operability in case of temporarily unavailable participants. Additionally, they are able to cope with failure due to frequent disconnections in order to still allow for successful commit. However, they are on the one hand not capable of integrating different structural cooperation patterns in one transaction. Furthermore, they all (as well as ATMs and flexible transactions) rely on statically defined structures of transactions.

In our work, we aim at loosely coupling of components. We abandon the separation of working- and commit-phase, as we try not to limit the autonomy of participants by tightly coupling their execution to rigorous transaction phases. Only in the worst case, in which loosely coupling endangers atomic outcome, we hark back on commit protocols.

### 3.2.2 Composition of Services

In our work, we aim at transactional correctness of ad-hoc cooperations implemented as composite services. We therefore in the following relate our work to research regarding transactional service composition and flexible workflows.

#### 3.2.2.1 Transactional Composition

**Protocols and Composition Operators** The following protocols and service composition operators have been proposed to support transactional composition. Most of them have been standardized for the use of Web Services.
The Transaction Internet Protocol\(^3\) (TIP) is a transport protocol which enables distributed coordinators to communicate via the Internet. It employs 2PC to achieve atomic outcome. Additionally, it specifies waiting periods as recovery measures in case the communication between participants fails.

Similar to WS-Tx, the Business Transaction Protocol (BTP) [OAS02] is designed to provide transactional support for loosely coupled cooperative business processes and to overcome the shortcomings of TIP (blocking of resources) [Pap03]. It is an XML based protocol which specifies the messages to be exchanged between participants and coordinators of a transaction. It aims at orchestrating loosely coupled web services into a single business transaction. For a detailed comparison between BTP and WS-Tx, we refer to [LF03].

The Tentative Hold Protocol (THP) as specified by the W3C [THP01b, THP01a] defines a framework to exchange messages prior to the actual transaction. It allows for tentative, non-blocking reservations (holds) of resources. If a client confirms a resource, other holders are notified of the expiration of their reservations. A combination of THP with compensation and negotiation methods prior to the transaction phase is proposed by [LY04]. The main benefits from employing tentative holds as introduced by THP are the up-to-date knowledge of availability of resources as well as reducing the number of cancelations due to unavailability of items [Pap03].

All of the above mentioned approaches share with ATMs and the cited mobile transaction models that they tightly couple services to transactions. Likewise, their definition of the compositions remains static. They all define message to be exchanged between participants and coordinators however are not able to spontaneously integrate forward recovery by exploring dynamically discovered services at runtime.

The approaches cited in the following were published more recently. They all propose the use of \textbf{composition operators} to transactionally support composite services.

In [LHL06], Liu et al. propose a composition operator which takes dependencies between services into account which may describe various relationships in order to assure correctness. These dependencies are similar to the ones employed by Gaaloul et al. [GRGH07] which we further investigate in Section 3.2.3. The proposed operator evaluates the quality of service of a composite service according to response time and execution costs of the components. This approach is extended in [LL07], by considering temporal constraints of services to ensure termination of composite services.

Fauvet et al. [FDDB05] make use of the THP for their high level operator for composing Web Services according to transactional properties. Services are required to implement THP and are distinguished according to their additional capabilities: Support of 2PC, compensatability or neither. This approach is interesting as it explores the capabilities of participants (i.e., the protocols they implement) present at runtime.

---

\(^3\)TIP is defined in RFC2371 [EKL98]
3.2 Related Research Areas

However, as opposed to our work, it still tightly couples services to transactions. Furthermore, as we integrate our approach into existing standards, we are able to map structural requirements (i.e., the control flow) to a single transaction as well.

Transactional Workflow Management  Additionally to the just presented protocols and operators, there exist approaches which propose comprehensive concepts to transactionally support the composition and execution of workflows. We subsume these in the following referring to them as transactional workflow management approaches. We briefly present the following transactional workflow management systems which are representative for other existing approaches, such as TSME [GHKM94], FlowMark [Ley95] and others [HvRR07, VV04, BDSN02].

A long-running activity as introduced by Dayal et al. [DHL91] recursively consists of activities and transactions. Control and data flow is separately defined in the activity’s script, either statically or by event-condition-action rules. The model includes means for compensation and communication, especially querying the status of an activity.

The METEOR model [KS95, RS95] integrates many aspects of earlier ATMs and workflow management systems. A workflow consists of a set of tasks which are arranged in well-defined structures. Conditions trigger transitions between tasks. In [MSKW96, SKM+96], METEOR is extended by failure recovery and error handling for distributed heterogeneous environments. METEOR guarantees failure atomicity, that is either an committed acceptable termination state is reached or a defined accepted aborted states in case of failure.

The aim of the Exotica project [AMA+95, AAA+96] is to explore advanced transaction models in workflow contexts. As already briefly stated in Section 3.2.1.1, the authors conclude, that ATMs only partly solve the problem of providing transactional support for workflow systems. This relies among others on the fact that workflow systems integrate process and user oriented concepts which are beyond the scope of ATMs.

These approaches have in common (with each other as well as the ATMs) that they on the one hand rely on static definition of transactions (as well as failure handling) at design time. They do not enable binding of dynamic participants and respectively adapting transactions to the current execution context at runtime. As opposed to our approach, they focus on specification at design time and verification at runtime.

Failure handling through forward-recovery and thus dynamic service binding at runtime has been proposed, e.g. by [SDN07]. This approach makes use of an abstract service provider which is responsible for replacing failing services with semantically equivalent ones. However, this concept limits itself to technically support one failure situation. As opposed to this, we ensure transactional execution of the whole workflow at runtime.

---

4 Please note, that the distinction between transactional workflow management systems and ATMs is not strict. Some approaches may be differently classified in the literature.
More recently, frameworks to assist service composition in P2P environments are developed. In [HMR07], a model focusing on transactional composition in P2P environments is proposed. It employs a decentralized serialization graph to enable global concurrency control. However, transactional support in the sense of failure atomicity is not given.

### 3.2.2.2 Flexible Workflows

The research area of *dynamic* service composition has received much interest in the past decade. It involves the task of synthesizing entirely new workflows by composing services to achieve overall goals [SK03, MS02, PWSK07]. Many of these approaches rely on AI planning techniques which, given an initial state, seek for sequence of operations to reach a defined goal state [GNT04]. As opposed to these, we do not regard completely automated composition from scratch rather than adapting cooperations defined as workflows with dynamic participants.

In the following, we relate our work to approaches which we classify as *flexible workflows* as they alter existing definitions of workflows at runtime. We distinguish between work that deals with *concepts* of finding provisioning strategies and those, which *technically* enable the alteration of a BPEL process at runtime.

The Web Service Management System (WSMS) as introduced by Srivastava et al. [SMWM06] enables querying multiple Web Services through an SQL-like interface. It regards selectivity and response time of Web Services to minimize the overall costs of an execution plan. More generally, Stein et al. [SPJ09, Ste07] adopt flexible provisioning of Web Services to maximize the profit of workflows and reduce costs. They provide several provisioning strategies regarding utility costs and failure probability to achieve an optimized composition [Ste08].

Similar to these approaches is the work published in the research area of workflow scheduling. It identifies the problem of finding correct execution sequences for workflow activities, obeying inherent constraints, e.g. temporal or causality constraints [ASSR93, DKRR96]. This is especially interesting in mobile environments considering the limited power constraints of heterogeneous devices [GB01]. Furthermore, some scheduling approaches focus on minimizing communication costs or ensuring prearranged QoS obligations defined in service level agreements [DD07].

We classify our approach as a conceptual approach employing flexible workflows well as. However, we complement the referenced work by providing transactional guarantees for flexible compositions.

Additionally to these conceptual strategies, there exist approaches which provide means to technically alter deployed BPEL processes at runtime. In [LLM+08] for example, a management framework for BPEL is presented. By introducing an additional

---

5 We omit the presentation of general frameworks which do not expose transactional support. The interested reader is referred to [MBB+03], for a comparison of these.
abstraction layer, the authors propose interoperable management of BPEL processes. This resource based management framework further enables manipulation of process models and instances. Such a framework builds the technical foundation for our approach of flexibly altering workflows at runtime.

### 3.2.3 Verification of Composite Services

Along with the research area of automated service composition, the challenge of verification of composition has accordingly gained interest. We classify such approaches according to verification criteria and (as it is closer related to our work) focus on those, aiming at transactional correctness.

#### 3.2.3.1 Transactional Correctness

In [GRGH07], an event calculus is proposed to verify transactional correctness in the sense of failure atomicity. As it is based on similar prerequisites as ours, we present this approach in depth.

The authors define a *transactional service* as a triple, consisting of an ID, a set of states and transitions. Transitions between states are either internally triggered (i.e., by the service itself) or externally, e.g., by another service or a person. The set of states varies according to the transactional properties of a service: Services are classified to be *compensatable, retrievable* or a *pivot*. The authors state, that a service may naturally combine certain properties [BPG05, BGP06].

For occurring events, the predicates *Happens, Initiates, Terminates* and *HoldsAt* are defined for certain points in time. These are used to perform transitions between states, i.e., an event or the termination of an event trigger a certain transition. Additionally, these are employed to define *transactional consistency rules*, which imply the type of failure handling that is allowed for each workflow pattern. E.g., the abortion of an active services as failure handling is only allowed for a co-occurring execution of both services.

A *transactional composite service* is then defined as a tuple, consisting of a set of component services and then the set of predicates which specify the external transitions of the components. This is done by means of dependencies which model the control flow as well as failure handling of the composite service.

By means of the consistency rules, a transactional composite service as specified by the designer is validated prior to runtime. Thereby, invalid failure handling is detected and the designer is notified. Additionally, consistency rules are used to determine whether the composite service terminated consistently after the execution. As a result, potentially missing failure handling may be detected [GBH⁺07]. In this case, the designer is notified and the transactional composite service accordingly evolves.

This formalism is very powerful and in our opinion well-suited to specify transactional behavior of composite services. Therefore, we base our formalism on this approach.
3.2 Related Research Areas

The framework, which implements the event-based calculus, is as already stated, focused on verifying user defined failure handling and detecting occurring inconsistencies after execution respectively. As opposed to this, we argue that adding failure handling may be automated by a system according to service properties and control flow patterns. Despite detailed distinction of failure recovery mechanisms, this approach does not ensure transactional guarantees for the whole workflow. For example, coordination of several pivot services in one composite service in order to guarantee consistent outcome is not incorporated.

Moreover, our work differs from the presented approach, as we aim at verification and adaptation of a composite service before inconsistent system states occur. Furthermore, we optimize existing compositions in the current execution context in terms of autonomy of elements, i.e., we minimize the number of elements which need to be coordinated via a blocking commit protocol.

3.2.3.2 Further Correctness Criteria

Additionally to the work which enables verification regarding transactional support in the sense of failure atomicity, there exist numerous models which validate compositions according to diverse correctness criteria. Many of them are based on petri nets as they enable characterization of composite service behavior in a broad range.

Challenges of composite service verification often relate to obstacles typically occurring in parallel processing: Response time determination and deadlock recognition are for example addressed by [BFHS03], as well as [HB03, NM02].

Similar in the formalism, however different in terms of correctness criteria are the approaches presented by [CFP02] and [MPP02]. Both model orchestration of services across organizational boundaries. Thus, typical challenges include negotiation and fulfillment of contractual agreements as well as assigning respective process responsibilities in process evolutions.

The work introduced in [BCGP08] addresses verification regarding security issues: The authors model messages and message exchange during execution of workflows using DAML-S and petri nets. They aim at checking integrity and authentication requirements. They provide, among others, the possibility to verify that messages sent have not been modified by third parties.

As opposed to these approaches, we verify whether specifications composite services fulfill the transactional demands in the sense of atomic outcome. We thereby complement the mentioned work, in order to enable successful ad-hoc cooperation in heterogeneous environments. We base our work on the formalism described by [GRGH07] as it is powerful yet simple to capture transactional behavior and has thus proven to be suitable for our challenges.
3.3 Summary

We identified several areas which are all related to our approach in different aspects. In summary, the differences of the presented approaches to our work and thus their unsuitability of transactionally supporting ad-hoc cooperations are due to the following reasons:

Foremost, the difference of almost all cited approaches dealing with transactional correctness (a.o., ATMs, flexible transactions and mobile transaction models) is the *incapability of flexibly reacting to the current execution context*. They mostly incorporate methods for creating transactions at design time, however limit themselves to verification of statically designed transactions (and failure handling) at runtime. Thus, in case verification fails, they are not able to explore and accommodate to the requirements and capabilities of dynamically discovered participants.

Additionally, many approaches offering transactional support *tightly couple* their participants to transaction phases (i.e., working- and commit-phases), e.g., WS-AT and some mobile transaction models. Thereby, the *autonomy of participants is restricted* as their execution relies on a coordinators decision. Approaches supporting autonomy of participants (e.g., WS-BA, SAGAS, ConTract), on the other hand rely on the assumption that all services are compensatable. These practices share with e.g. failure handling enabled by BPEL or [SDN07], that they do not provide any verification means, thus *no transactional correctness guarantees* can be given.

Further, we identified the *inflexibility to incorporate different structural patterns* into a single transaction, *different correctness criteria* (e.g., global serializability in flexible transactions, or resource constraints flexible workflows) or generally *no transactional support* (e.g., utility costs in dynamic service composition) as crucial differences to our approach.

Based on these differences of the referenced work to our approach, we believe that they are not at all or only partially suitable to transactionally support ad-hoc cooperations in heterogeneous environments.
4 Discovering Mobile Services

We identified the capability to discover services at runtime as a prerequisite for ad-hoc collaboration, since entities are only able to cooperate if they are able to find others. As stated in Section 2, we assume participants to be equipped with several network interfaces. The underlying networking infrastructure therefore changes with the chosen communication channels: Nodes may communicate via fix basestations (potentially costly, reliable) or directly in an ad-hoc manner (no monetary costs, volatile).

If fix basestations exist, well-established service discovery algorithms for distributed systems exist (e.g., UDDI or SLP [Net97]). These mostly employ designated repository nodes, which are responsible for storing advertisements. The suitability of these approaches for ad-hoc networks is to a limited extent as they rely on the availability of repository nodes. There exist protocols for ad-hoc scenarios, which try to reduce the effects of mobility and the volatile communication infrastructure by still offering highly available service information in the network. As many mobile devices are nowadays equipped with positioning sensors (e.g., GPS), we utilize position information of nodes and exploit their mobility. Thereby, our approach adapts to the current movement of nodes and reduces the number of messages if possible.

In this chapter, we briefly classify existing protocols for mobile ad-hoc networks and additionally present our approach which explicitly exploits the mobility of nodes to provide accurate information.\footnote{As the focus of this thesis lies on transactional support of ad-hoc collaboration, we keep the presentation short and refer to corresponding literature.}

4.1 Existing Approaches

Generally, mobile service discovery protocols are distinguished according the maintenance of their information: There exist proactive techniques, in which service offers are actively distributed within the network and reactive protocols in which advertisements are locally stored and service requests are distributed. Höpfner et al. [HTKR05] further classify protocols to the following categories:²

Centralized Approaches  In centralized approaches, one or several designated repository nodes exist at which providers register their services. Providers proactively advertise

²For a detailed survey of service discovery protocols we refer to [MPHS05, MBB09].
4.1 Existing Approaches

their offers to one or several directory nodes. Clients search for suitable services at these repositories. A well-known representative of these approaches is the Service Location Protocol (SLP) [Net97]. A variant of SLP for MANETs is proposed in [Pen05]. The performance of these approaches relies on the availability of single nodes which is the main drawback of them in MANETs.

**Flooding**  A simple yet efficient way of discovering services in volatile environments is based on flooding. Either advertisements (proactively) or service requests (reactively) are spread in the network. Depending on the approach, matching of requests and advertisements is done on the client or the provider side. Representatives of such protocols are e.g., the Simple Service Discovery Protocol (SSDP) [Int] and JXTA-Search [Wat01]. While information on services is highly available, the produced network load is immense.

**Hash-Based Approaches**  Hash-based protocols employ mathematical hash functions to transform service descriptions and service requests into numerical values. These values correspond to physical addresses of devices present in the network. Advertisements and requests are forwarded to the address obtained by the hash-function. This node is responsible for matching requests to the service. The Content Addressable Network (CAN) [RFH+01] as well as Tapestry [ZKJ01] fall into this category of protocols. As hash functions randomly map values, semantical closeness of descriptions does not result in physical closeness of devices. This may hinder the discovery process as requests hardly ever exactly match the description of the provider.

**Overlay-based Protocols**  Many service discovery protocols for MANETs rely on maintaining overlay structures in the network. These overlays are formed according to semantic design decisions, e.g., [KKRO03b, KKRO03a], or semantic-free criteria such as topological closeness, e.g., [RFH+01, CZH+99, SBR04]. Information about available services in the respective cluster is aggregated. According to this information, a node decides whether to forward a request to nodes in its cluster or in a different cluster. The performance of these approaches is strongly influenced by the message overhead produced by maintaining the overlay structures.

**Semantic Routing**  As opposed to simple flooding techniques, nodes distribute information to participants in their environment. According to this information, service requests are selectively forwarded to nodes which are more likely to match the request in their surrounding. Among others, representatives of this kind of protocols are Group-based Service Discovery (GSD) [CJFY02, CJFY06], Konark [HDVL03], CNPGSDP [GWYY06] and Allia [RCJF02].
4.2 Adaptive Group-based Service Discovery: aGSD

In this section, we propose our service discovery algorithm aGSD which is designed to exploit the mobility of participants. Our approach is a group-based approach. We first give an introduction into the concepts of group-based service discovery (similar to [CJFY06, GWYY06]) and then present how we apply this concept to take advantage of users’ mobility.

4.2.1 Group-based Service Discovery

Group-based service discovery is a hybrid approach to service discovery in which both, service offers and service requests, are distributed to a certain degree. All nodes maintain a service cache to store received advertisements. Any peer matches received requests with the offers stored in its local cache. The core idea of group-based service discovery is to semantically classify services according to a previously defined hierarchy. This classification is utilized to selectively forward requests to nodes which are more likely to store an according offer. An example of such a hierarchy is shown in Figure 4.1.

![Hierarchical grouping of services.](image)

*Example:* Reconsider our running example, illustrated in Figure 1.2 on page 4. The Transportation service matches the categories Software, Ticketing and Transportation. The Confirm service which confirms the offers and prints the according tickets on the other hand is classified as Hardware and Printer.

If services are classified according to the predefined hierarchy, advertising services, caching offers locally and searching for appropriate offers is performed as follows.

**Advertising Services** Any provider wanting to advertise its services, initiates a service advertisement message regularly which contains the following information:

```plaintext
< adID, providerAddress, serviceDescription, serviceGroups, cachedGroups, lifetime, numberOfHops, pos_SP, mov_SP, t_SP >
```
4.2 Adaptive Group-based Service Discovery: aGSD

Such an offer is uniquely identified by its \textit{adID}. The \textit{providerAddress} denotes the address of the provider of the offered service, whose actual description is encoded in the \textit{serviceDescription}. The provider also sends a list of groups that its service belongs to according to the offered functionality (\textit{serviceGroups}, see Figure 4.1). Using \textit{cachedGroups} the provider sends a synopsis of its local cache: This list contains all groups of offers which the provider currently stores in its cache. This information is later on used to selectively forward a request to the according nodes. The \textit{lifeTime} of the service identifies the time it should be kept within the cache. This is related to the rate in which a service provider advertises its services (advertisement frequency \( f_{adv} \)). The \textit{numberOfHops} denotes the number of hops the offer is supposed to be spread. Additionally, the current position and movement of the service provider is encoded: \( pos_{sp} \) identifies the nodes position, \( mov_{sp} \) the providers direction and speed (as a vector) at time \( t_{sp} \). This corresponds to the information stored when monitoring moving objects, such as proposed in [SWCD97] and adopted by [HBS+06].

\textbf{Caching Advertisements} Any node receiving such an offer inspects the \textit{adID} to check whether it has already received it. Upon first retrieval of an advertisement, it stores it in its local cache. A \textit{cache entry} of a node consists of the following information, which directly refers to the information of advertisements (as presented above):

\[
< \text{adID}, \text{providerAddress}, \text{serviceDescription}, \\
\text{serviceGroups}, \text{cachedGroups}, \text{lifeTime} >
\]

The caching node proceeds according to the number of hops the advertisement is supposed to be spread. If the \textit{numberOfHops} is greater than one, the node decrements it, updates the \textit{cachedGroups} of its own cache, and broadcasts the advertisement. Otherwise, the extent to which the offer is supposed to be propagated is reached.

The caches of nodes are renewed according to the lifetime of the offers: If the lifetime is expired, the advertisement is deleted. If the cache’s capacity is exceeded, offers are replaced according to a least-lifetime strategy.

\textbf{Searching for Services} A node in search of a service tries at first to find a matching advertisement in its local cache. If it cannot locate an appropriate advertisement, it creates a \textit{service request} with the following parameters:

\[
< \text{serviceDescription}, \text{requestGroups}, \\
\text{requestAddress}, \text{hopCount}, \text{maxHopCount} >
\]

The \textit{serviceDescription} contains the description of the desired service. \textit{requestGroups} is a list of groups to which the service belongs to. The \textit{requestAddress} is the address of the requesting node. \textit{hopCount} denotes the current number of hops the advertisement
4.2 Adaptive Group-based Service Discovery: aGSD

has already traveled, thus it is initialized with 0. maxHopCount encodes the maximum number of hops, the requesting node wants its request to be propagated.

Each node receiving a request proceeds as follows: If it cannot be locally matched and hopCount has not yet exceeded maxHopCount, it increments hopCount and re-transmits the request. In order to do so, the node checks the cachedGroups of its cache entries. If any match the requestGroups, it forwards the request to the matching nodes according to the providerAddress of the entries. If no match can be found, the request is simply broadcasted.

4.2.2 Exploitation of Mobility

As introduced so far, aGSD complies with conventional group-based service discovery algorithms, such as [CJFY02, GWYY06]. We designed aGSD to make use of position- and mobility information in two ways: On the one hand, the mobility information of nodes is used to dynamically configure and thus adapt the protocol to the current context. On the other hand, we use position information to enable discovery of remote services.

4.2.2.1 Dynamic Configuration

The performance of the protocol is mainly determined by the rate at which service providers advertise their services (advertisement frequency \( f_{adv} \)) and the lifetime of offers \( l \). They heavily influence the performance of the approach in terms of network load (costs) and discoverability of services (benefits). Thus, there is a trade-off between the number of messages to be sent and the availability of service advertisements. If, as proposed by [CJFY06], the configuration is user controlled, it is somewhat arbitrary to decide for settings of \( f_{adv} \) and \( l \): Mis-calibration leads to dissatisfactory performance results. Moreover, advertisement frequency and lifetime of corresponding offers should be related to each other, as otherwise, the following may happen:

- If the advertisement frequency and the lifetime of the advertisements is fairly high, nodes being in direct transmission range of the provider receive new offers although a corresponding entry still exists in their cache. Nodes having moved out of the direct vicinity of the provider keep the entry even though they cannot directly contact the provider anymore.

- If opposed, the advertisement frequency and the lifetime of the advertisements are low then the availability of the service advertisements decreases. Nodes in the vicinity delete the corresponding cache entry before a new one is sent.

On account of these simple considerations, we configure the advertisement frequency of providers and the lifetime of offers as follows:
Adaptation of the advertisement frequency to the mobility of the service provider

We base the adaptation of the advertisement frequency to the movement of the service provider on the following assumption: If it moves rapidly, its set of neighboring nodes changes quickly. In order to keep the availability of the advertisements high, the advertisement frequency has to be high. If on the other hand the providers position is stable, messages are saved by decreasing $f_{adv}$ and therefore increasing the length of the interval until the next advertisement is sent (denoted as $I_{adv}$). As stated above, the lifetime of advertisements is chosen according to the advertisement frequency of the provider.

We calculate the advertisement rate according to the movement of the node: Whenever the distance to the last position at which it sent an advertisement is estimated to be greater than a given threshold, a new advertisement is sent. We denote this threshold value in terms of the broadcast range of the service provider. This leads to the following computation of the advertisement frequency $f_{adv}$ (and the length of the advertisement interval $I_{adv}$ respectively):

$$I_{adv} = \frac{1}{f_{adv}} = \frac{cF \times r_{bc}}{|mov_{sp}|} \quad (4.1)$$

$I_{adv}$ is the length of the interval until the next advertisement is sent, $r_{bc}$ the broadcast range, $|mov_{sp}|$ the velocity of the service provider and $c_F$ an application parameter that determines the threshold value for the distance in terms of the broadcast ranges. The calculated length of the advertisement interval is bounded to $[I_{min}, I_{max}]$. Thus, we avoid too many advertisements to be sent when moving fast and still enforcing advertisements to be sent when the node hardly moves at all.

The lifetime of the advertisements calculated by the service provider (denoted $l_{sp}$) is chosen in relation to the advertisement frequency. We calculate the lifetime as:

$$l_{sp} = c_L \times I_{adv}, \quad \text{with } c_L \geq 1 \quad (4.2)$$

Parameter $c_L$ denotes the relation of $l_{sp}$ and $f_{adv}$. By doing so, aGSD adapts to the movement of the service provider: If it moves fast, the availability of its advertisements increases as it sends more advertisement messages. If the provider moves slowly, it saves messages by still keeping the availability high.

Adaptation of the lifetime of cache entries according to the mobility of nodes

By calculating the lifetime of a service offer according to the advertisement frequency, the cache renewal is already adapted to the mobility of the service provider. Advertisements of stable (and slow) service providers have greater lifetimes as opposed to those of fast moving providers. Upon calculation of the lifetime, the service provider is only aware of its own movement. A node receiving the offer is additionally able to analyze his mobility and thus consider its relative movement to the provider.
By considering the relative movement to the service provider, a node decreases the lifetime of an advertisement if their distance is predicted to drastically increase. We consider the estimated distance \( d(t) \) between a node caching an advertisement and the service provider to adjust the lifetime of service offers. The distance between these two nodes as a function of time is calculated by the euclidean distance (see Figure 4.2):

\[
d(t) = |(pos_{sp} + mov_{sp} \times (t - t_{sp})) - (pos_{n} + mov_{n} \times (t - t_{n}))|
\] (4.3)

for \( t > t_{sp} \) and \( t > t_{n} \). The receiving node utilizes the mobility information of the service provider which is provided in the advertisement (\( pos_{sp}, mov_{sp} \) and \( t_{sp} \)) and its own mobility, denoted by \( pos_{n}, mov_{n} \) and \( t_{n} \). If the distance of \( sp \) and \( n \) exceeds a computed threshold within the lifetime of the advertisement, the lifetime is lowered. The threshold is determined according to the \( numberOfHops \) of the offer and the broadcasting range of the node \( r_{bc} \). The point in time at which the distance exceeds the threshold is denoted as \( t_{x} \). The node calculates \( t_{x} \) as the solution of the following equation:

\[
d(t + t_{x}) = numberOfHops \times r_{bc}
\] (4.4)

The receiving node sets the lifetime of the advertisement as the minimum of the lifetime computed by the service provider \( l_{sp} \) and the time \( t_{x} \) until the distance exceeds the threshold:\(^3\)

\[
l_{n} = min(l_{sp}, c_{L} \times t_{x})
\] (4.5)

Note that a node only lowers the lifetime of an entry within its cache but does not increase it. If a node’s movement is similar to the one of the service provider, it receives a new advertisement according the calculated advertisement frequency which is before \( l_{sp} \) (given by the service provider) has expired. Therefore increasing the lifetime of a cache entry does not increase the performance of the algorithm.

\(^3\)Just as \( l_{sp}, t_{x} \) is also multiplied by the scaling factor \( c_{L} \).
All in all, aGSD explores mobility information of nodes to decrease the lifetime of advertisements whenever nodes move away from each other. Thereby, service requests which are initiated due to outdated cache entries are spared and the expenses, i.e., number of messages per successful services usage, are reduced.

4.2.2.2 Remote Service Discovery

In ad-hoc environments, nodes are usually interested in services of providers in their direct vicinity. Corresponding requests are fairly well answered by existing approaches (including aGSD as introduced so far). However, requests, which do not refer to the immediate surrounding of a node, can hardly (or not at all) be handled. Such requests are certainly useful in our application scenario MoP. For example, searching for services which are in the proximity of a certain point of interest the user is approaching, such as the Brandenburg Gate (see Figure 4.3). We refer to requesting such service as remote services in the following.

Figure 4.3: Searching for remote services.

The concept of location based services (LBS) [VS04] seems at first closely related to remote services. However, LBS usually refer to the immediate surrounding of a client as opposed to remote services which assume the provider to be in a defined target area. Existing conventional service discovery protocols do not support remote service requests. They do not integrate forwarding service requests according to geographical criteria. Forwarding a request to nodes in a certain target area is therefore impossible. Employing the mobility information of nodes, aGSD enables such remote service requests (and thus remote service usage). This is done by employing geographical routing of requests.

In order to do, the requests of aGSD are enhanced by the following parameters:

< requestType, pos_target, radius_target >

The requestType encodes, whether the request is a remote request (REMOTE) or a conventional search (LOCAL). If the search is local, possibly specified values of the target region are disregarded. Otherwise, the request is forwarded to the target region which is specified by its plane coordinates (pos_target) and its radius (radius_target).

If a remote request is posed, the protocol proceeds as follows: The query is geographically routed to nodes in the target area. In our implementation of the protocol, we
employ GPSR (Greedy Perimeter Stateless Routing) [KK00a, SWLF04] to geographically forward requests to the according area. If a node within the target region receives the request, it processes it just as a local request: If it is able to match the request to an offer stored in its local cache, it generates an according response. Otherwise, it forwards the request to its neighbors according to the group information in its cache. If a node in the target area matches the request, it as well employs geographical routing to re-route the response back to the initiator. If the request cannot be routed to the target region (e.g., if there are no nodes in the target area), it is dropped if the maximum hop count is reached.

By utilizing the mobility information of nodes, aGSD enables the discovery and usage of services in a specified target area.

4.3 Evaluation

Additionally, to the implementation of aGSD, we experimentally evaluated the protocol in order to prove its suitability for ad-hoc scenarios. After introducing the setup, we present results for the dynamic configuration and the remote service discovery. As mentioned above, service discovery is not the focus of this thesis, we limit the presentation to selective results. For a detailed evaluation, we refer to [Bie08].

4.3.1 Evaluation Setup

The vital points of interest when evaluating aGSD are its suitability and efficiency in ad-hoc situations. We therefore simulated aGSD in ad-hoc scenarios using the ns-2\(^4\) network simulator. Our objectives are to show, that the up-to-dateness of advertisements is increased (benefits) and the number of messages per successful service usage is reduced (expenses). Additionally, we want to demonstrate, that by exploiting mobility information, aGSD enables discovery and usage of remote services. In summary, in the context of this thesis, we want to show, that aGSD enables mobile services to be found and used in ad-hoc scenarios.

4.3.1.1 Configurations

We compare the following configurations of the protocol in order to evaluate the influence of the exploitation of mobility information.

\textbf{GSD} is the statically configured: The advertisement frequency is \(f_{adv} = 1/15\text{s}\), i.e., each service provider sends an advertisement every 15s (\(I_{adv} = 15\text{s}\)).

\(^4\)http://www.isi.edu/nsnam/ns/
aGSD is dynamically configured: The advertisement interval $I_{\text{adv}}$ is however bound to $[I_{\text{min}}, I_{\text{max}}] = [10s, 30s]$. The inherent protocol parameter $c_F$ is set to $c_F = 1$ (according to the results of [Bie08]), while $c_L$ is varied in the shown experiments.

aGSD' is as well dynamically configured, however in this configuration only the lifetime of offers is adapted according to the relative movement of a caching node and the provider. $f_{\text{adv}}$ is statically configured to $f_{\text{adv}} = 1/15s$.

### 4.3.1.2 Metrics

In order to quantify and compare the efficiency of the protocol configurations, we consider the following metrics.\(^5\)

**Request hits** The request hits indicate the overall number of cached advertisements that are matched to all service requests during the simulation.

**Packet Loss** The packet loss indicates the ratio of messages which get lost during the simulation. For example, the invocation of a service may be lost, if it is initiated due to an outdated advertisement and the provider is not available anymore.

**Expenses** The expenses of a configuration indicate the number of messages that the protocol initiated during the simulation in relation to the number of successful service invocations.

### 4.3.2 Evaluating the Dynamic Configuration

**Set-Up** In order to evaluate the dynamic configuration of aGSD, we simulated the following scenario: 50 nodes move on a 200x200m\(^2\) area following the Random Waypoint Mobility Model [JM96]. The simulation time is 600s. The velocity of nodes ranges from $v_{\text{min}} = 1m/s$ to $v_{\text{max}} = 3m/s$; their broadcasting range is 30m. The duration of the so-called thinking-time of nodes is 5s.

Each nodes offers 10% of all services. Service requests for a randomly chosen service are initiated regularly every 5s on a randomly chosen node. If its request is matched, the client invokes the discovered service.

All shown evaluation results represent average values of at least 100 simulation runs.

**Evaluation** In Figure 4.4 on the left hand side, the number of request hits of the three configurations GSD, aGSD and aGSD’ are shown (on the y-axis), varying $c_L$ on the x-axis.\(^6\) GSD achieves more request hits than aGSD which and aGSD’ for all values of $c_L$. It is obvious that the number of hits increases with ascending values of $c_L$, as with $c_L$ the lifetime of offers accordingly increases.

---

\(^5\)In our experiments, we regarded further metrics, e.g., up-to-dateness and false-positives which are not explicitly evaluated in this thesis. We therefore forego their presentation.

\(^6\)Recall, $c_L$ defines the relation of $I_{\text{adv}}$ and $l_{\text{sp}}$ (and $l_{\text{n}}$ respectively).
However, the number of request hits does not indicate the overall performance of the configurations: The rate of successful service usage is roughly 60% for all three configurations (variations between them is lower than 5%). Therefore, considering the packet loss in this scenario, depicted on the right hand side of Figure 4.4, it easily becomes apparent, that the packet loss of GSD is roughly 5% to 10% greater than aGSD. The packet loss of aGSD’ is even slightly lower than for aGSD. This relies in the number of service invocations which are initiated in consequence of service hits to providers which are not available anymore: In this case, the cache entry of matching nodes are outdated and thus lead to messages being lost.

Therefore, GSD produces the highest expenses of all: Although GSD achieves the greatest number of request hits, it produces considerably higher packet loss ratios than aGSD and aGSD’. This relies in the fact, that many of the request hits are outdated.

In Figure 4.5, we varied the maximum velocity \( v_{\text{max}} \) of nodes between \( v_{\text{max}} = 3\, \text{m/s} \) and \( v_{\text{max}} = 19\, \text{m/s} \). In this set-up, the configuration parameter \( c_L \) is set to \( c_L = 3 \). The greater the maximum velocity of nodes, the more offers are actually sent using aGSD while \( f_{\text{adv}} \) remains fixed for GSD and aGSD’. In this scenario, the number of request hits (see Figure 4.5 on the left hand side) is nearly constant for GSD and aGSD and decreasing for aGSD’ with ascending \( v_{\text{max}} \). This relies on the fact, that aGSD’ decreases the lifetime of offers with increasing \( v_{\text{max}} \), however as opposed to aGSD does not produce more advertisements.

The ratio of successful service usage equally decreases from roughly 66% (\( v_{\text{max}} = 3\, \text{m/s} \)) to ca. 45% (\( v_{\text{max}} = 19\, \text{m/s} \)) for all configurations. (The ratio of successful service usage is slightly greater for aGSD than for the other two protocols.) Considering the packet loss, depicted in Figure 4.5 on the right side, it becomes apparent, that the packet loss of all protocols increases enormously: While it is almost doubled for GSD (26% to 48%) and aGSD (17% to 38%), the increase for aGSD’ is lower (15% to 24%).

This results in the overall expenses as depicted in Figure 4.6: The expenses for GSD
increase, as outdated information about service providers lead to increased package loss. The expenses for \(aGSD\) evolve even worse with increasing \(v_{max}\): The information, nodes keep in their cache when using \(aGSD\) is more up-to-date; however the costs produced by many more advertisements are not justifiable, as in this scenario the rate of successful service usage decreases. The reasons for that rely in the scenario: Due to the density of nodes, \(aGSD\) has no trouble discovering services, however with a maximum velocity of 19m/s, the usage of services often fails.

As opposed to these two configuration, \(aGSD'\) is able to slightly reduce the expenses. By reducing the lifetime of offers, fewer request hits are achieved. However, as these achieved already consider the relative movement of nodes, they are up-to-date. Thereby, in this case \(aGSD'\) produces convincing results.

**Conclusion** The evaluation of the dynamic configuration shows, that service discovery benefits from exploring mobility information to enable discovery and usage in ad-hoc scenarios. In ca. 60% of all request, one or more providers (which may be invoked) are
discovered. Especially, exploiting the *relative movement* of nodes in the configuration \( aGSD' \) is convincing: In comparison to \( GSD \) and \( aGSD \), the expenses for successful service usage are considerably decreased, as invocations to unavailable providers are avoided. Regarding the protocol parameters, \( c_F = 1 \) and \( c_L \in [2.5, 3.5] \) achieve the best results regarding the expenses. The protocol in the configuration \( aGSD' \) outperforms the other two configurations.

We are thereby able to show that exploration of mobility information may beneficially used for service discovery in ad-hoc scenarios.

### 4.3.3 Evaluation of Remote Service Requests

**Test Set-Up**  As opposed to the previous series of test, we alter the scenario as follows: 10 designated service providers are additionally placed diagonally in a specified target area. These nodes do not move. Periodically, a node is chosen randomly to initiate a remote service request to one of these target areas. The radius of the target area is set to 15\( m \). As stated before, we use GPSR in case requests are to be geographically routed. The results represent evaluation of roughly 28 000 remote service requests.

**Evaluation**  This protocol feature of \( aGSD \) performs quite well. Ca. 65\% of remote service requests are replied. In 91\% of these cases, the service invocation is successful; in 94\% of these, the output of the service is successfully returned. The overall success rate is therefore roughly 55\%.

![Distance to provider for discovery and usage of remote service.](image)

Figure 4.7: Distance to provider for discovery and usage of remote service.

Considering Figure 4.7, it becomes apparent that the average distance of clients to the target area when *discovering* remote services successfully (on the right) is \( \sim 75m \). The average distance of clients to the target area which successfully *used* services is slightly lower with 72\( m \). Considering unsuccessful discovery, the average distance is 102\( m \) and 99\( m \) for unsuccessful service usage. Obviously, the farther away a node from the target area, the likelier its request or its invocation is not successful.
In Figure 4.8, the correlation of the distance between the searching node and the target area and the distance between the searching node and the service provider is depicted. It easily becomes apparent, that these distances are closely related, i.e., the response to the request is initiated in the specified target area. Additionally, the figure shows, that in this set-up considerably more successful usages are initiated for distances lower than 100m. However, discovering and usage of remote services is successful up to $\sim 250m$ distance between provider and client.

**Conclusion** In conclusion, the feature of remote service recovery which is enables by exploring mobility information of nodes, performs well in the evaluated scenario. By employing geographical routing, $aGSD$ enables remote service discovery at a great distance (72m average, in single cases up to 250m).

### 4.4 Summary

In this chapter, we proposed our service discovery algorithm $aGSD$ for ad-hoc scenarios. $aGSD$ is a group-based approach which exploits the mobility of nodes to adapt to the current situation and enable remote service requests. Our experimental evaluations show, that especially the analysis of relative movement of nodes positively affects the protocol: $aGSD$ reduces the lifetime of advertisement of nodes, if they do not linger in their communication ranges anymore. Thereby, the overall expenses of the protocol are noticeably reduced. Furthermore, $aGSD$ enables the use of remote services: Mobility information of nodes is utilized to geographically forward requests to certain destination. Thereby, one is able to pose requests to a certain target region which is of great interest in our target scenario MoP. Our evaluations results show, that this feature performs quite well in the simulated scenario.
5 Formalizing Transactional Cooperation of Services

In this chapter, we introduce the formal model used to specify transactional behavior of services and composite services. Using our formal model, we abstract from the characteristics (e.g., mobility) of single components. As it is very powerful and suits our demands, we base our model on that of Bhiri et al. [BPG05, BGP06] and their event-algebra [GBH+07, GRGH07]. We extend it by a classification of services according to more relevant service properties and defining these properties for patterns. Additionally, we specify transactional correctness in the presence of these transactional properties.

5.1 Transactional Services

5.1.1 Service Model

In order to model the autonomous (internal) behavior of a single service we employ the state/transitions model of [GRGH07]. Figure 5.1 shows the state-machine for a single service: Before invocation, its state is *initial*. After being activated, it is *active*. *Failed* and *canceled* indicate failed execution i.e., no changes are made persistent, either due to an internal error (*failed*) or externally triggered (*canceled*). If the service completes successfully, it transits to the *completed* state.

The transitions between these states are either internally or externally triggered. Internal transitions (indicated by solid lines in Figure 5.1) are completed due to events generated by the service itself (e.g., completion). External transitions (indicated by dashed lines) are triggered by another entity, such as the workflow engine or a human.

![State-machine of a single service.](image)

To indicate the completion status of a service $s$, we employ a boolean representation...
5.1 Transactional Services

to encode whether the service is completed: I.e., $s$ indicates, that the service completed (that is its changes are made persistent). If the service is failed, canceled, compensated or has not been activated, we denote its status by $\neg s$.

According to a service’s transactional properties, additional states or transitions are added. These are introduced in the next section.

5.1.2 Transactional Properties of Services

To ensure atomic outcome of composite services the characteristics of services have been considered in transaction management in SOAs in the past, as well as the characteristics of subtransactions especially in heterogeneous multidatabase (MDBS) environments. In the following, we use the terms subtransaction and service interchangeably.

Generally, existing classifications either distinguish transactions according to the protocols they implement (e.g., [FDDB05, LY04]). Or the approaches abstract from precise protocols and classify subtransactions according to transactional properties without relying on specific implementations, e.g., in MDBS [MRKS92b, MRKS92a, BGMS92, EJK+96, ZNBB94, ZNB01, DU96]. These were later on adopted for transaction models for composite services, among others by [VV04, MBB+03, BPG05, BGP06, GRGH07, GBH07].

These approaches mostly employ the same terms, however define them partially different. E.g., [BPG05] state that the effects of a pivot subtransaction cannot be undone. However, it indeed can be retrievable. That contradicts the definition of [BGMS92].

In order to overcome this obstacle and after exhaustive study of existing business transaction models, we regard the following transactional service properties as boolean properties, which we first published in [HS08]: Compensatability, consistent closure and retrieability. We employ the approach of using transactional properties as opposed to implemented protocols, as this suits the abstraction level of the formal model.

---

**Definition 1. Compensatability of a Service $s$**

A service $s$ is defined to be compensatable if there exists an available service $c$ which semantically undoes the effects of $s$. The compensatability of $s$ is denoted as:

\[
s.\text{compensatable} = 1
\]

If a service $s'$ is non-compensatable, it is denoted as $s'.\text{compensatable} = 0$.

**Example:** Consider our running example on page 4, a booked ticket to the Philharmonics may be compensated by canceling the booking.

---

1See Chapter 3 for further discussion.
through a compensate-transition and a compensated-state (see Figure 5.1). Compensatability indicates, whether the effects of a service can be undone after completion.

In our system model, we assume the compensation service $c$ of a service $s$ to be available. Thus, if a provider specifies its service to be compensatable, it ensures, that the compensating service $c$ can be invoked via a reliable communication channel.

Through the inclusion of a service in the workflow, a designer states, whether the completion of the service is inevitable for the completion of the workflow. It is vice versa assumed, that a service is only allowed to be completed if the workflow is completed. E.g., no hotel room is allowed to be booked, if the whole trip is not booked. However, some services may allow for inconsistent completion i.e., they complete although the workflow may be canceled. This is usually given by the consistency requirements of the data the service operates on or the semantics of the service.

We therefore define the following transactional service property, which – to the best of our knowledge – has not been considered for transactional service composition before.

**Definition 2. Consistent Completion of a Service $s$**

A service $s$ is defined to demand consistent completion (or consistent closure) with respect to the outcome of the whole workflow, if it needs, once completed, recovery in case of failure. Its completion infers the completion of the whole workflow. This property of $s$ is denoted as:

$$s\text{.consistentCompletion} = 1$$

If $s$ does not demand consistent closure (denoted as $s\text{.consistentCompletion} = 0$) does not need to be compensated in case of backward-recovery.

*Example:* Consider the following example of reserving a hotel room: If the hotel states that the reservation of a room is automatically canceled unless it is explicitly confirmed at least a week before the check-in day, this reservation service does not demand consistent closure. If the whole trip is canceled, the reservation will be automatically deleted, as it is not confirmed. Another example is the `Confirm` service of our running example.

Other examples of services, which do not demand consistent closure, are read-only participants of transactions as specified by the WS-Tx. The significance of the consistent closure property conveys whether a service needs to be compensated in case of failure.

The third property, according to which services are classified, is retrievability which is defined as follows.
5.1 Transactional Services

Definition 3. Retrievability of a Service $s$
A service $s$ is defined to be retrievable (or redoable), if it will eventually complete, if its activation is repeated in case of failure. The retrievability of a service $s$ is denoted as:

$$s.\text{retrievable} = 1$$

If a service $s'$ is not retrievable, it is denoted as $s'.\text{retrievable} = 0$.

The property states, whether the execution of the service can fail. This is an important feature of the above mentioned compensating service: Assuming the compensatability of a service, it is also assumed, that the compensating action will complete (i.e., not fail). Redoability of a service is modeled through a redo-transition (see Figure 5.1).

In our system model, a mobile service provider which specifies its service to be retrievable, ensures to be available via a reliable network channel. I.e., if no ad-hoc communication can be established, it can still be invoked using reliable network channels.

We define the complete transactional properties\(^2\) of a service as follows:

Definition 4. Complete Transactional Properties of a Service $s$ ($pCT(s)$)
The complete transactional properties of a service $s$ are defined as the following triple of properties:

$$pCT(s) := (s.\text{compensatable}, s.\text{consistentCompletion}, s.\text{retrievable})$$

A service may hold an arbitrary combination of the complete transactional properties as defined in Definition 4. Therefore, $2^3 = 8$ different types of services are possible.

However, not every property is important for all purposes. For example, for verification purposes (see Section 6.2), it is important to know whether a service is redoable and whether its completion hinders the correctness of the workflow in case of failure. Thus, it is important to know, whether a service is compensatable or does not need consistent closure. We denote this by the derived property recoverability as follows:

Definition 5. Recoverability of a Service $s$
A service $s$ is defined to be recoverable, if it is either compensatable or it does not demand consistent closure, thus:

$$s.\text{recoverable} := (s.\text{compensatable} = 1) \lor (s.\text{consistentCompletion} = 0)$$

\(^2\)Please note, that completeness of these properties is only given in the scope of this thesis. We do not claim universal completeness of our defined properties.
A service $s$ which is recoverable is accordingly denoted as:

$$s.\text{recoverable} := 1$$

If a service $s'$ is not recoverable, it is denoted as $s'.\text{recoverable} = 0$.

Regarding the service model introduced in Section 5.1.1, such a service is modeled just as a compensatable service: If the service is compensatable, it holds a compensated state and a compensate transition. If the service does not demand consistent completion, it is also modeled using a compensate transition and the compensated state.

Regarding this derived property, we define the derived transactional properties (or transactional properties for short) as a tuple as follows:

**Definition 6. Derived Transactional Properties of a Service $s$ ($p_T(s)$)**

The derived transactional properties (transactional properties for short) are defined as the tuple of properties recoverability and retrieability. They are denoted as:

$$p_T(s) := (s.\text{recoverable}, s.\text{retrievable})$$

**Example:** In Figure 5.2, the running example with (complete and derived) transactional properties of services is shown. As CRS is retrieable and does not need consistent closure (i.e., it does not need to be compensated in case of failure) its complete properties are $p_{CT}(CRS) = (0, 0, 1)$, thus its derived properties are $p_T(CRS) = (1, 1)$. PayCC on the other hand, which offers payment per credit card, is compensatable and needs consistent closure. As it is not retrieable, its complete transactional properties are $p_{CT}(PayCC) = (1, 1, 0)$, thus its derived properties are $p_T(PayCC) = (1, 0)$.³

![Figure 5.2: Running example with transactional properties.](image)

³We employ the *-symbol as a wildcard, if a service exposes a certain property or not. That is, if a service $s$ exposes the properties $p_T(s) = (1, *)$ it is recoverable, however it may or may not be retrievable.
For analyzing and flexibly adapting the composite service at runtime (as it is done in Chapter 6), it is sufficient to regard the derived transactional properties. In order to add the correct and entire failure handling (see Section 5.2.5.3), the complete transactional properties are needed.

5.2 Transactional Composition of Services

5.2.1 Control Flow Patterns

Besides services, control flow patterns (or workflow patterns alternatively) are elementary components of workflows. As their name indicates, their task is to specify the flow of control. Thereby, they define the structure of the composition.

Originally, they were defined by van der Aalst, ter Hofstede et al. [WAB00, AHKB03, ABEWO0]. The Workflow Management Coalition (WfMC) [Coa99] defined the following simple control flow patterns: Sequence, parallel split, synchronization, exclusive choice and simple merge (see Appendix A.1).

The separation of split and join patterns allows for the specification of sophisticated workflows in which forked branches are not merged. For the purpose of transactional workflow patterns, we specify each split pattern to be ended by a matching join pattern, at the end of the workflow at the latest. As most workflow specification languages (e.g., BPEL) are block-oriented, this complies with the standards for implementing a workflow. A block surrounds its elements employing distinctly labeled begin- and endpoints.

We employ \( WP(E) \) to denote a workflow patterns with a set of elements \( E \). An element \( e \in E \) is either a service or a workflow pattern. Thereby, patterns are recursively defined. We use an index to distinguish between the types of pattern. Additionally, we specify the ATS-status of patterns.\(^4\) Recall, by \( s \) we denote the successful completion and of service \( s \) while \( \neg s \) indicates, that \( s \) is not completed. The ATS-status of a given pattern \( WP(E) \) is a boolean expression which specifies the accepted termination states of \( WP(E) \), i.e., the state of the included elements for which the pattern is completed.

**SEQUENCE** \( WP_{SEQ}(e_1, \ldots, e_n) \)\(^5\) defines for all of its elements to be sequentially executed, thus \( e_i \) is activated after the completion of \( e_{i-1} \). We assume the elements \( e_1, \ldots, e_n \) to be arranged in increasing order of their index. Thus, as soon as \( e_n \) completes, the sequence is considered to be completed. Its ATS-status is \( e_1 \land e_2 \land \ldots \land e_n \).

**AND** \( WP_{AND}(e_1, \ldots, e_n) \) groups the parallel split and the synchronization. It determines, that all elements \( e_i \in \{e_1, \ldots, e_n\} \) are executed in parallel. At the join point, they are synchronized: The subsequent workflow is activated as soon as all branches completed, thus its ATS-status is denoted as \( e_1 \land e_2 \land \ldots \land e_n \).

\(^4\)Recall, that ATS refers to the committed acceptable termination states.

\(^5\) \( WP_{SEQ} \) complies with the sequence pattern as defined by the WfMC.
5.2 Transactional Composition of Services

**XOR** \( WP_{XOR}(e_1, \ldots, e_n) \) subsumes the exclusive choice and the simple merge. Based on a mechanism, one \( e_i \in \{e_1, \ldots, e_n\} \) is chosen. The subsequent workflow continues as soon as one element completes, thus one and only one element \( e_i \in \{e_1, \ldots, e_n\} \) is completed. Successful execution of the pattern, i.e., its ATS-status, is expressed by \((e_1 \land \neg e_2 \land \ldots \land \neg e_n) \lor (\neg e_1 \land e_2 \land \neg e_3 \ldots \land \neg e_n) \lor \ldots \lor (\neg e_1 \land \ldots \land \neg e_{n-1} \land e_n)\).

Note, that in case of \( WP_{XOR}(E) \), we only regard patterns, in which the choice of branches is done according transactional properties of services. The elements of \( WP_{XOR}(E) \) are therefore considered to be alternatives.

**Example:** Consider again our running example as depicted on page 4. Abbreviating Philharmonics, Transportation and Reservation service by the respective first character, the workflow is composed as follows:

\[
WP_{SEQ}(CRS, WP_{AND}(P, T, R), \text{Confirm}, WP_{XOR}(\text{PayCC}, \text{PayCh}))
\]

Other advanced patterns as defined by the WfMC are disregarded in the scope of this thesis: They can either be composed by simple patterns or do not specify deterministic or transactional behavior (see Appendix A.2).

5.2.2 Transactional Pattern

According to the defined transactional properties of elements, there exist situations in which aligning elements in different patterns is not sufficient to guarantee correct execution. In these situations, we coordinate elements using blocking protocols, such as for example 2PC (e.g., by employing WS-AT, see Section 3.1.1.1).

Therefore, we define an additional auxiliary pattern, the **Subtransaction pattern** as:

**Definition 7. Subtransaction Pattern** \( WP_{subTA}(e) \)

The *subtransaction pattern*, denoted as \( WP_{subTA}(e) \), contains exactly one single control flow pattern \( e \). The type of \( e \) defines the control flow of the enclosed elements. \( WP_{subTA}(e) \) defines, that all elements \( e' \) contained by \( e \) have to be coordinated in an atomic subtransaction.

Elements which are enclosed in a *Subtransaction pattern* \( WP_{subTA} \), can no longer be autonomously executed as introduced in Section 5.1.1, but their execution has to be enrolled to a transaction.\(^6\) Elements of the subtransaction are coordinated by a coordinator in order to guarantee atomic output of all of them. The subtransaction pattern is thus orthogonal to the defined control flow patterns as it does not define the control flow of the enclosed elements.

\(^6\)Solitary exception are *indirect conflict elements*, which are introduced in Section 6.2.1.3.
5.2 Transactional Composition of Services

5.2.3 Transactional Properties of Patterns

In order to analyze the workflow at different levels of abstraction, we define the transactional properties of workflow patterns. They are determined according to the type of the pattern and the properties of the contained elements. In the following, we define the derived transactional properties (recoverability and retrievability) for all patterns. Specification of the complete transactional properties can be found in the Appendix A.3.

The definition of SEQUENCE and AND both define correct execution to be that either all of their elements have to be completed, or all of their elements, which need to be compensated, have to be compensated. Therefore, for the sake of simplicity, the definition of their transactional properties is commonly presented.

**Definition 8. Transactional Properties of \( WP_{SEQ}(E) \) and \( WP_{AND}(E) \)**

The transactional properties of a pattern \( WP(E) \) containing a set of elements \( E \), which is a \( WP_{SEQ}(E) \) or an \( WP_{AND}(E) \) pattern are defined as the following tuple:

\[
p_T(WP(E)) := (WP(E).recoverable, WP(E).retrievable)
\]

where \( WP(E) \) is

- **recoverable**, if all elements are recoverable
  \[
  WP(E).recoverable = 1 \\
  \iff \forall e \in E : e.recoverable = 1
  \]

- **retrievable**, if all elements are retrievable
  \[
  WP(E).retrievable = 1 \\
  \iff \forall e \in E : e.retrievable = 1
  \]

As the enclosed elements of an \( WP_{XOR} \) pattern are alternatives, the properties for \( WP_{XOR}(E) \) are defined differently. One and only one element is to be completed (and to be compensated accordingly). If several alternatives exist, the transactional properties of \( WP_{XOR}(E) \) cannot always be definitely determined beforehand. In these cases, the definite properties are derived during execution. Handling these uncertainties is described in 6.3.3.

**Definition 9. Transactional Properties of \( WP_{XOR}(E) \)**

The transactional properties of XOR pattern \( WP_{XOR}(E) \) are defined as:

\[
p_T(WP_{XOR}(E)) := (WP_{XOR}(E).recoverable, WP(E).retrievable)
\]

where \( WP_{XOR}(E) \) is
**5.2 Transactional Composition of Services**

- **recoverable** if all elements are recoverable. If none of the elements is recoverable, \( WP_{\text{XOR}}(E) \) is not recoverable.

  \[
  WP_{\text{XOR}}(E)_{\text{recoverable}} = 1 \\
  :\iff \forall e \in E : e.\text{recoverable} = 1 \\
  WP_{\text{XOR}}(E)_{\text{recoverable}} = 0 \\
  :\iff \forall e \in E : e.\text{recoverable} = 0
  \]

- **retrieable**, as soon as one element is retrieable.

  \[
  WP_{\text{XOR}}(E)_{\text{retrieable}} = 1 \\
  :\iff \exists e \in E : e.\text{retrieable} = 1
  \]

As soon as one element \( e \) in the pattern is retrieable, the whole pattern is retrieable. If the retrieable element is executed, it is guaranteed to complete. Thus, the whole pattern is guaranteed to complete.

The \( WP_{\text{subTA}} \) pattern is by definition not recoverable and not retrieable:

**Definition 10. Transactional Properties of \( WP_{\text{subTA}}(e) \)**

The transactional properties of a subtransaction pattern \( WP_{\text{subTA}}(e) \) are defined as:

\[
 p_T(WP_{\text{subTA}}(e)) := (WP_{\text{subTA}}(e).\text{recoverable}, WP_{\text{subTA}}(e).\text{retrieable})
\]

with \( WP_{\text{subTA}}(e).\text{recoverable} := 0 \) and \( WP_{\text{subTA}}(e).\text{retrieable} := 0 \).

**Example:** In Figure 5.3, the running example workflow is depicted. The \( WP_{\text{XOR}} \) pattern is both recoverable and retrieable \( (p_T(WP_{\text{XOR}}(\text{PayCC}, \text{PayCh})) = (1, 1)) \) as both elements are recoverable and there exists a retrieable element \( (\text{PayCh}) \). The \( WP_{\text{AND}} \) pattern is neither recoverable nor retrieable \( (p_T(WP_{\text{AND}}(P, T, R)) = (0, 0)) \), as it comprises non recoverable and non retrieable elements. The encompassing \( WP_{\text{SEQ}} \) pattern, which consists of four elements \( (\text{CRS}, WP_{\text{AND}}(P, T, R), \text{Confirm} \text{ and } WP_{\text{XOR}}(\text{PayCC}, \text{PayCh})), \) is for the same reason not recoverable and not retrieable \( (p_T(WP_{\text{SEQ}}(E)) = (0, 0)) \).

**5.2.4 Workflow Elements**

As stated in our system model, we specify ad-hoc cooperations as composite services and thus implement them as workflows. By defining transactional properties of patterns, we are able to analyze a composite service at different levels of abstraction. The structure of a composite service and thus a workflow is **recursively** given by its elements:
5.2 Transactional Composition of Services

**Figure 5.3:** Properties of patterns of the running example.

**Definition 11. Workflow Elements $e$**

A service is a workflow element. Further, if $e_1, \ldots, e_n$ with $n > 1$ are workflow elements,

- $WP_{SEQ}(e_1, \ldots, e_n)$, $WP_{AND}(e_1, \ldots, e_n)$ and $WP_{XOR}(e_1, \ldots, e_n)$ are workflow elements;
- $WP_{subTA}(e_1)$ is also a workflow element.

Elements in the formal model are the equivalent of components in our system model. For our algorithms introduced in Chapter 6, we define the notion of **mandatory workflow elements**. These are elements, which need to be executed in order for the workflow $\omega$ to complete, thus no alternatives exist.

**Definition 12. Mandatory Workflow Element $e$ of Workflow $\omega$**

An element $e$ is defined to be a mandatory element of $\omega$, if it is an element of $\omega$ and not element of any $WP_{XOR}$ pattern in $\omega$.

By referring to mandatory workflow elements, we abstract from single alternatives in a workflow and regard the enclosing $WP_{XOR}$ pattern as an element itself.

**5.2.5 Dependencies**

A composite service consists of a set of elements and the definition of the relations between these elements. The relations include on the one hand data dependencies between elements, on the other hand behavioral relation of elements in the regular case (normal execution dependencies) and in case of failure (failure recovery dependencies).
5.2.5.1 Data dependencies

If parts of the output data of a workflow element are required as input data for another element, they are said to be data dependent on each other as follows:

**Definition 13. Data Dependency** $e_i \rightarrow e_j$

If the output data of workflow element $e_i$ is required as input data of workflow element $e_j$, $e_j$ is directly data dependent on $e_i$, denoted as $e_i \rightarrow e_j$.

If further, element $e_k$ is data dependent on $e_j$ (thus $e_j \rightarrow e_k$) however its input data is not directly dependent on $e_i$’s output data, $e_k$ is said to be transitively or indirectly data dependent on $e_i$: $e_i \rightarrow e_j \rightarrow e_k$.

*Example:* Recall our running example (see Figure 5.3 on page 46). Present data dependencies are depicted as gray dashed arrows: The confirmation of the bookings (Philharmonics, Transportation, and Reservation) can only be pursued, if the bookings are finished. Otherwise, the price cannot be determined and the data which the customer is supposed to confirm cannot be identified, thus: $\{P, T, R\} \rightarrow \text{Confirm}$. ▶

Data dependencies are given by the specification of a workflow. If $e_j$ is (in-)directly data dependent on $e_i$, $e_j$ cannot be invoked previously or concurrently to $e_i$. We assume data dependencies only to be given between elements, which are arranged in sequence. Elements of a $WP_{AND}$ or $WP_{XOR}$ patterns cannot be data dependent on each other.

5.2.5.2 Normal Execution Dependencies

Normal execution dependencies (denoted as $\text{depNrm}(e, e')$) specify the relation between workflow elements in case no failure occurs. Thus, they define the activation of an element $e'$ after the completion of the previous element $e$. These are implicitly given by the designer by arranging elements in workflow patterns.

5.2.5.3 Failure Recovery Dependencies

In addition to normal execution dependencies, failure recovery dependencies have to be defined, which specify the relation between workflow elements in case failure occurs. In our service model, the standard failure handling mechanisms are cancelation ($\text{Cln}$) of active elements, compensation ($\text{Cps}$) for completed elements or activation of alternatives ($\text{Alt}$). Events, which trigger these dependencies to be executed are failure ($\text{Fln}$) and cancelation ($\text{Cln}$) of an active element or compensation ($\text{Cps}$) of a completed element.
Therefore, we specify the following failure recovery dependencies:⁷

An alternative dependency \( \text{depAlt}(e, e') \) between elements \( e \) and \( e' \) specifies, that \( e' \) is activated in case \( e \) fails. The denotation of all other failure handling dependencies encodes the above mentioned causes and effects:

\[ \text{Example: For example, \( \text{depFlCl}\!n(e, E) \) specifies, that in case of failure of \( e \), all elements \( e' \in E \) are canceled.} \]

Cause of such a dependency to be triggered, can be the failure (\( \text{depFl}^* \))⁸, cancelation (\( \text{depCl}^* \)) or compensation (\( \text{depCps}^* \)) of a workflow element. The measures to be taken are either activation of other elements, cancelation of currently active elements (\( \text{dep}^* \text{Cl}\!n\text{-dependencies} \)), or compensation of previously completed (\( \text{dep}^* \text{Cps}\text{-dependencies} \)) elements.⁹

Up to now, it is the designers responsibility to add appropriate failure handling dependencies. As these are primarily dependent on the requirements of a service and the enclosed context (i.e., the workflow element), we argue that enhancing the workflow by failure handling dependencies can be automated to ensure correct execution. Adding appropriate dependencies is tedious, however straightforward. We refer to [HS09a] for how to append failure recovery dependencies. In the next section, we introduce our employed notion of correctness.

### 5.3 Specifying Correctness: Semi-Atomicity

Intuitively, the execution of the workflow is correct, if the workflow is completed (cf. commit). I.e., all services which need to be completed in order for the workflow to complete, are successfully finished and all others must not be completed. Transactionally correct execution also involves the situation in which the workflow is aborted (i.e., unsuccessfully finished) and all services which need to be compensated, are compensated.

This notion of correctness, including commit or abort, is closely related to strict atomicity of database transactions. However, investigating the requirements of services and the workflow, strict atomicity does not fit this scenario, as the structural demands of the workflow (e.g., choices through XOR-patterns) as well as transactional properties of services (e.g., services that do not demand consistent closure) are not met.

[ZNBB94] defined semi-atomicity in the context of flexible transactions (cf. Chapter 3). They explore some transactional properties of subtransactions as well as structural demands (employing the precedence and preference relation). We adapt this model of semi-atomicity and extend it to comprise our defined transactional service properties.

---

⁷Note, that similarly defined dependencies as in [GRGH07] are sometimes referred to transactional execution dependencies. However, we feel the term failure recovery dependencies is more appropriate.

⁸The *-symbol is used as the wildcard for all possible effects (and causes respectively) in the following.

⁹A definition of these dependencies is given in [HS09a].
5.3 Specifying Correctness: Semi-Atomicity

We want to emphasize, that our notion of correctness refers to a *terminated* workflow, thus its state *after* is has been executed. As stated in our system model (see Chapter 2), we employ the notion of *accepted termination states* (ATS), to specify successful completion of a workflow.

By defining the workflow, a designer implicitly defines ATS, thus representational sets of services whose completion represent the successful execution of the workflow. Note that multiple sets exists, as alternatives or multiple ATS may exist (for example *PayCC* and *PayCh* in our running example). We define semi-atomic termination (or semi-atomicity for short) of a workflow as follows:

**Definition 14. Semi-atomic Termination of a Workflow ω**

Semi-atomic termination (or semi-atomicity for short) of an executed composite service implemented as a workflow ω with specified ATS is defined as

- either all elements e belonging to one valid execution path to an ATS are completed and no other element e’ in ω demanding consistent completion (i.e., \( p_{CT}(e') = (\ast, 1, \ast) \)) is completed (i.e., their state is initial, failed, canceled or compensated)
- or no element e in ω demanding consistent completion (\( p_{CT}(e) = (\ast, 1, \ast) \)) is completed.

**Example:** Consider our running example as depicted in Figure 5.4: Shading of services imply their completion, while no shading implies that their state is initial, canceled, failed or compensated. Let the properties of *CRS* and *Philharmonics* allow for inconsistent completion (i.e., \( p_T(CRS) = p_T(P) = (\ast, 0, \ast) \)). The workflow is semi-atomically terminated (aborted), as no service demanding consistent completion is completed.

![Figure 5.4: Correct (unsuccessful) termination of the running example.](image-url)

This further relaxes correctness criterion defined for flexible transactions [ZNBB94]: It disregards backward-recovery for services which may complete inconsistently. We
employ this criterion to inspect all possible executions of a workflow. We are thereby able to verify, if a workflow will semi-atomically terminate in any case (see Section 6.2.2).
6 Flexible Workflows to Guarantee Correct Execution

Using the specified formal model of transactional workflows (see Chapter 5) and semi-atomicity as the correctness criterion for the terminated workflow, we introduce our algorithm to guarantee correct execution of a workflow. We therefore perform two basic steps (see Figure 6.1): At first the given workflow is verified in the current execution context by considering the transactional properties of its elements. If the verification fails, the control flow structure of the workflow is adapted prior to execution to ensure correctness. The workflow is again adapted during execution, in order to react to dynamic changes (i.e., failure of services and discovery of alternatives) which may occur in mobile environments (Section 6.5). Integrating service discovery in our adaptive workflow management in order to react to dynamic events is presented in Section 6.4.

6.1 Views of the Workflow

We introduce three different views of a workflow. These are not equipotent; each of them exemplifies different aspects of the workflow. The first one emphasizes the structure of
the composition (6.1.1), the second one exhibits the data dependencies (6.1.2) and the last one presents the accepted terminations states of the workflow (6.1.3).

### 6.1.1 Workflow as a Tree

For verification purposes, we analyze the structure of a workflow $\omega$ which is recursively given by its elements. We thus regard $\omega$ as a tree, denoted as $T_\omega$, as follows: Each element $e$ in $\omega$ corresponds to a node in $T_\omega$, labeled by its type (e.g., AND). The children $e'$ of $e$ in $T_\omega$ represent the elements $e'$ of $e$ in $\omega$. Note, that the children of nodes representing $WP_{SEQ}$ patterns have to be ordered according to their position in the pattern. For all other nodes, the order of children is irrelevant. Simply for clarity, we depict inner nodes as round nodes and shape leaf nodes rectangular.

![Diagram of tree view $T_\omega$ of the running example.]

**Example:** In Figure 6.2, the tree representation of our running example (recall Figure 5.2 on page 41) is depicted.

We employ this view to verify the workflow (Section 6.2). This depiction of the workflow allows for holistic examination at different layers of abstraction.

### 6.1.2 Data Dependency Graph $G_\omega(V, E)$

All existing data dependencies of a workflow $\omega$ are represented by its data dependency graph $G_\omega(V, E)$. It contains the mandatory elements (recall Definition 12) of the workflow and their data dependencies as follows:

**Definition 15. Data Dependency Graph $G_\omega(V, E)$**

The data dependency graph $G_\omega(V, E)$ is a directed acyclic graph. The set of vertices $V$ contains all mandatory elements $e$ of $\omega$. A directed edge $(v_i, v_j)$ between nodes $v_i$ and $v_j$ exists, if there is a direct data dependency in the form $v_i \rightarrow v_j$.

In practice, the specification of workflows (e.g., using BPEL) does not allow cyclic data dependencies. Therefore, the data dependency graph is acyclic. $G_\omega(V, D)$ contains
6.1 Views of the Workflow

all mandatory elements, as we abstract from single alternatives rather than holistically examine the set of alternatives.

Example: The data dependency graph of the running example (see Figure 5.3 on page 46) is illustrated in Figure 6.3.

![Data dependency graph](image)

Figure 6.3: Data dependency graph $D_\omega(V, E)$ of the running example.

Data dependencies have to be preserved when adapting the workflow in order to guarantee executability. We perform our adaptation algorithm (Section 6.3) on this view.

6.1.3 ATS View

The third view, which we employ, represents the state of elements of a workflow $\omega$ in case of successful execution. It is a boolean expression which specifies the accepted termination states of $\omega$.

**Definition 16. ATS View of the Workflow $ATS_\omega$**

Let $T_\omega$ be the tree view of workflow $\omega$. The ATS view of $T_\omega$ is defined to be the $ATS$-status of the root node of $T_\omega$: The $ATS$-status of the root of $T_\omega$ is derived by replacing each inner node in $T_\omega$ by the $ATS$-status of its subtree.

We already stated the ATS-status for single workflow patterns in Section 5.2.1. As a workflow $\omega$ is recursively composed by its elements, its ATS view is given by the $ATS$-status of its root node in $T_\omega$.

$$ATS_\omega = CRS \land (P \land T \land R) \land Confirm \land ((PCC \land \lnot PCh) \lor (\lnot PCC \land PCh))$$

![ATS view](image)

Figure 6.4: ATS view $ATS_\omega$ of the running example.
Example: In Figure 6.4, the ATS view of our running example is depicted.

This view on the workflow is used in order to verify, that the ATS-status of a converted workflow still infers the ATS-status of the original workflow. We therefore employ this representation to demonstrate the correctness of our approach (Section 6.3.2).

6.2 Verification of a Workflow

Objective of the verification is to a-priori validate whether the execution of a specified workflow is semi-atomic in any case. This is done according to the structure of the composition and the transactional properties of workflow elements. At first, we identify elements, which endanger semi-atomicity of an execution. They are referred to as conflicting elements. We explore their influence on the execution of a workflow and thereafter define the correctness criterion for a workflow: As opposed to semi-atomic termination, which defines the correctness for a termination of a workflow, the verification algorithm validates the correctness of a workflow before the executions, thus validate the correctness of all possible terminations.

6.2.1 Conflict Elements

In this section, we examine pairs of elements, which hinder the correct execution of the workflow. They are referred to as transactional conflict elements. Intuitively, these are pairs of elements, which are able to produce failure situations, which cannot be healed by backward- or forward-recovery.

6.2.1.1 Transactional Conflict Elements

The semi-atomicity of an executed workflow is not preserved, if an element, which is not recoverable (i.e., \( p_T(e) = (0, \ast) \)) is completed while another mandatory element, which is not retrievable (i.e., \( p_T(e) = (\ast, 0) \)) failed. We therefore define a pair of transactional conflict elements as follows:\(^1\):

\[
\text{Definition 17. Transactionally Conflicting Pair of Elements}
\]

A pair of mandatory workflow elements \( e_i \) and \( e_j \) is defined as a transactionally conflicting pair of elements, denoted as \( \{e_i, e_j\}_C \), if \( e_i \) is not recoverable and \( e_j \) is not redoable, i.e.:

\[
\begin{align*}
  p_T(e_i) &= (0, \ast) \\
  \land p_T(e_j) &= (\ast, 0)
\end{align*}
\]

\( \{e_i, e_j\}_C \) is also referred to as a transactional conflict.

\(^1\)Note, that this concept strongly differs from transactional conflicts in the context of transaction schedulers.
The curly brackets indicate, that the order of the elements in the enclosing workflow \( \omega \) is not regarded. However, the notation always implies \( e_i \) to be the non recoverable and \( e_j \) to be the non redoable element. If such a pair of elements exists within a workflow, they cannot be executed independently of each other.

**Example:** Assume for example \( e_1 \) to be not recoverable but redoable i.e., \( p_T(e_1) = (0, 1) \), and \( e_2 \) to be recoverable but not redoable i.e., \( p_T(e_2) = (1, 0) \), as shown in Figure 6.5. \( \{e_1, e_2\}_C \) is then a conflicting pair of elements. If they are executed concurrently (Figure 6.5.a), \( e_1 \) might complete while \( e_2 \) fails. Therefore, the execution is not semi-atomic i.e., not correct. The same holds, if they are executed in the order \( e_1 \) before \( e_2 \) i.e., not recoverable before not redoable, as shown in Figure 6.5.b.

If both elements are not recoverable and not redoable i.e., \( p_T(e_1) = p_T(e_2) = (0, 0) \) a subtransaction coordinating \( e_1 \) and \( e_2 \) (using the \( WP_{sub\,TA} \) pattern as introduced in Section 5.2.2), e.g. \( WP_{sub\,TA} (WP_{AND} (e_1, e_2)) \) cannot be avoided in order to guarantee semi-atomic execution.

![Figure 6.5: Rearranging transactional conflict \( \{e_1, e_2\}_C \) (a) in sequence (b) and (c).](image)

Otherwise, rearranging the order to executing the recoverable element before the retrievable element, their execution guarantees semi-atomic completion (Figure 6.5.c): \( e_1 \) is only invoked if \( e_2 \) previously completed. As \( e_1 \) is redoable, it will eventually complete. Therefore, this alteration of the workflow ensures semi-atomicity. However, this re-arrangement is only possible, if no data dependency between these elements exists.

### 6.2.1.2 Directed Transactional Conflict Elements

If a data dependency between a conflicting pair of elements exists in the following way, we refer to this as a *directed transactionally conflicting pair of elements*.

![Definition 18. Directed Transactionally Conflicting Pair of Elements](image)

A pair of workflow elements \( e_i \) and \( e_j \) is defined as a *directed transactionally conflicting pair of elements* \( (e_i, e_j)_C \), if transactional conflict \( \{e_i, e_j\}_C \) exists and \( e_j \) is (directly or indirectly) data dependent on \( e_i \), thus

\[
\{e_i, e_j\}_C
\]
6.2 Verification of a Workflow

\( \wedge \) it exists \( e_i \rightarrow e_j \)

\((e_i, e_j)_C\) is also referred to as a directed transactional conflict.

Note, that – as the term suggests – the order of the elements is given: If \((e_i, e_j)_C\) is a directed transactionally conflicting pair of elements, \((e_j, e_i)_C\) is not. Otherwise, cyclic dependencies in the form \(e_i \leftrightarrow e_j\) would exist. Directed transactional conflicts between pairs of elements cannot be solved by rearranging them as their order is fixed.

**Example:** Consider for example the elements \(e_1\) and \(e_2\) and their transactional properties depicted in Figure 6.6. According to the definition, a directed transactional conflict between \(e_1\) and \(e_2\) exists i.e., \((e_1, e_2)_C\). In this case, semi-atomicity can only be preserved by utilizing a subtransaction pattern \(WP_{subTA}\) and altering the workflow to be \(WP_{subTA}(WP_{SEQ}(e_1, e_2))\). Thereby, the autonomy of these elements is decreased.

\[\text{Figure 6.6: Directed transactional conflict } \{e_1, e_2\}_C \text{ (a), enclosed in } WP_{subTA} \text{ (b).}\]

Considering directed transactionally conflicting elements more closely, it is straightforward to conclude the following proposition:

**Proposition 1. Transitivity of Conflicts**

Assume \((e_i, e_j)_C\) and \((e_j, e_k)_C\) to be directed transactional conflicts. It holds, that

a. \(p_T(e_j) = (0, 0)\) and

b. \((e_i, e_k)_C\) is also a directed transactional conflict.

**Proof.** Assume \((e_i, e_j)_C\) and \((e_j, e_k)_C\) to be directed transactional conflicts.

The proof is straightforward using Definition 18.

Claim 1: \(p_T(e_j) = (0, 0)\)

- Since \((e_i, e_j)_C\) are directed transactionally conflicting, \(e_j\) is not retrievable (following the definition), thus \(p_T(e_j) = (\ast, 0)\). Since \((e_j, e_k)_C\) are directed transactionally conflicting, \(e_j\) is neither recoverable \((p_T(e_j) = (0, \ast))\). Thus, the transactional properties of \(e_j\) are \(p_T(e_j) = (0, 0)\).
6.2 Verification of a Workflow

Claim 2: \((e_i, e_k)_C\) is a directed transactionally conflicting pair.

- According to the definition of transactionally conflicting elements, and the existing conflicts \((e_i, e_j)_C\) and \((e_j, e_k)_C\), \(e_i\) is not recoverable \(p_T(e_i) = (0, \ast)\) and \(e_k\) is not redoable \(p_T(e_k) = (\ast, 0)\). Additionally, data dependencies \(e_i \rightarrow e_j\) and \(e_j \rightarrow e_k\) exist. Therefore, an indirect data dependency \(e_i \rightarrow e_j \rightarrow e_k\) exists. It follows, that \((e_i, e_k)_C\) is a directed transactionally conflicting pair.

Example: Consider for example elements \(e_1\), \(e_2\) and \(e_3\) as depicted in Figure 6.7. Since directed transactional conflicts \((e_1, e_2)_C\) and \((e_2, e_3)_C\) exist, \(p_T(e_1) = (0, \ast)\) and \(p_T(e_3) = (\ast, 0)\). Additionally, \(e_3\) is (at least transitive) data dependent on \(e_1\) thus \(e_1\) and \(e_3\) form as well a directed transactional conflict \((e_1, e_3)_C\).

\[
\begin{array}{ccc}
(0, \ast) & (0, 0) & (\ast, 0) \\
\text{e1} & \text{e2} & \text{e3}
\end{array}
\]

Figure 6.7: Transitive conflict \(\{e_1, e_3\}_C\).

The transitivity of conflict elements concludes directed transactional conflicts between elements with direct data dependencies to be given. It is also worthwhile to examine the characteristics of elements which are enclosed (concerning data dependencies) by a directed transactionally conflicting pair of elements.

**Proposition 2. Enclosed Conflict Elements**

Assume data dependencies \(e_i \rightarrow e_j\) and \(e_j \rightarrow e_k\) to exist as well as the directed transactional conflict \((e_i, e_k)_C\). It then holds that:

a. \((e_i, e_j)_C \iff p_T(e_j) = (\ast, 0)\)

b. \((e_j, e_k)_C \iff p_T(e_j) = (0, \ast)\)

c. \(e_j\) neither conflicts with \(e_i\) nor \(e_j \iff p_T(e_j) = (1, 1)\)

Proof. Assume data dependencies \(e_i \rightarrow e_j\) and \(e_j \rightarrow e_k\) and the directed transactional conflict \((e_i, e_k)_C\) to exist. The proof is again straightforward using Definition 18.

Claim a: \((e_i, e_j)_C \iff p_T(e_j) = (\ast, 0)\)
6.2 Verification of a Workflow

- ‘⇐’: Assume $p_T(e_j) = (\ast, 0)$. Because the directed transactional conflict $(e_i, e_k)_C$ exists, $e_i$ is not recoverable $p_T(e_i) = (0, \ast)$. Since $e_j$ is data dependent on $e_i (e_i \rightarrow e_j)$, it follows that $e_i$ and $e_j$ are a directed transactionally conflicting pair $(e_i, e_j)_C$.

- ‘⇒’: Assume $(e_i, e_j)_C$ to directed transactionally conflict. According to Definition 18, it follows, that $p_T(e_j) = (\ast, 0)$

Claim b: $(e_j, e_k)_C \iff p_T(e_j) = (0, \ast)$, proof is analogue to proof of claim 1.

- ‘⇐’: Assume $p_T(e_j) = (0, \ast)$. Because the directed transactional conflict $(e_i, e_k)_C$ exists, $e_k$ is not redoable $p_T(e_k) = (\ast, 0)$. Since $e_j$ is data dependent on $e_j (e_j \rightarrow e_k)$, it follows that $e_j$ and $e_k$ are a directed transactionally conflicting pair $(e_j, e_k)_C$.

- ‘⇒’: Assume $(e_j, e_k)_C$ to be a directed transactional conflict. According to Definition 18, it follows, that $p_T(e_j) = (0, \ast)$.

Claim c: $e_j$ neither conflicts with $e_i$ nor $e_k \iff p_T(e_j) = (1, 1)$

- ‘⇐’: Assume $p_T(e_j) = (1, 1)$. According to Definition 18, $e_j$ cannot be element of directed transactionally conflicting pair.

- ‘⇒’: Assume $e_j$ to neither conflict with $e_i$ nor $e_k$. According to Proposition 2.1 and 2.2, it then follows that $p_T(e_j) \neq (0, \ast)$ and $p_T(e_j) \neq (\ast, 0)$. Thus $p_T(e_j) = (1, 1)$.

In Figure 6.8, the cases stated in Proposition 2 are illustrated. In Figure 6.8.a, $e_2$ is not retrieable, thus it conflicts with $e_1$: $(e_1, e_2)_C$. On the other hand, if $p(e_2) = (0, \ast)$ as depicted in Figure 6.8.b, $e_2$ conflicts with $e_3$, thus $(e_2, e_3)_C$ exists. In Figure 6.8.c, $e_2$ is recoverable and retrieable, thus it neither conflicts with $e_1$ nor $e_3$.

![Figure 6.8: Enclosed conflict element forms conflicts (a,b), is not part of conflict (c).](image)

6.2.1.3 Indirect Conflict Elements

Proposition 2 conveys, that there exist elements, which are enclosed by a directed transactionally conflicting pair of elements concerning the existing data dependencies, however do not conflict with the enclosing elements. We refer to such elements as indirect conflict elements.
Definition 19. Indirect Conflict Elements
An workflow element $e_j$ is defined as an indirect conflict element, if it is recoverable and retrievable, and it is enclosed by a directed transactional conflict $(e_i, e_k)_C$, thus
\[ p_T(e_j) = (1, 1), \]
\[ \land \ (e_i, e_k)_C \text{ exists, and} \]
\[ \land \ e_i \rightarrow e_j \text{ and } e_j \rightarrow e_k \text{ exist.} \]
Note, that indirect conflict element $e_j$ neither conflicts with $e_i$ nor $e_j$ (following Proposition 2).

Example: Consider the example depicted in Figure 6.8.c: The directed transactional conflict $(e_1, e_3)_C$ exists. $e_3$ is data dependent on $e_2$ which in turn is data dependent on $e_1$ (illustrated by the dashed lines). Due to these data dependencies, re-arranging the order of this sequence is not possible. Semi-atomic execution of these can only be ensured, if $e_1$ and $e_3$ are coordinated in an subtransaction $WP_{\text{subTA}}$. In order to preserve the data dependencies, $e_2$ has to be executed in between these two elements. However, according to its properties ($p_T(e_2) = (1, 1)$), it can be executed autonomously i.e., it does not need to be coordinated using a blocking commit protocol (e.g., 2PC).

This generally holds for all indirect conflict elements: As they are guaranteed to complete and can be recovered in case of failure, they do not need to be coordinated.

As indirect conflict elements are always enclosed by conflicts and always employ the transactional properties $p_T(e) = (1, 1)$ they can be easily identified within a workflow. We therefore forego an explicit denotation.

6.2.2 Verification Criterion: SAP
Intuitively, a workflow is correct, if all possible executions result in semi-atomic termination. Thus, semi-atomicity of the workflow is preserved, if in case of failure of any included service,

- the execution of the workflow can be either backward-recovered (thus all previously completed elements are recovered)
- or there exists at least one alternate execution path to an ATS which is guaranteed to complete i.e., it is redoable.

We define this property, which we refer to as semi-atomicity preservation SAP as the correctness criterion for the verification of a workflow $\omega$. The criterion SAP denotes, whether the execution of a workflow element will in any case result in semi-atomic commit or abort.
Definition 20. Semi-Atomicity Preserving SAP
An element $e$ is SAP, if all possible executions of $e$ result in semi-atomic termination.

SAP of elements $e$ can be determined by considering every failure situation that might occur. If all of these situations can be recovered, by canceling active services, compensating for completed services or executing an alternative, $e$ is SAP.

Remark 1. SAP of a Single Service $e$
A single service $s$ is always SAP, as all of its termination states (completed, canceled, failed, compensated) are semi-atomic terminations of $s$.

When regarding workflow patterns, the combinations of failed and completed elements determines whether the pattern is SAP or not.

SAP of a Sequence Patterns

Example: Consider the workflow $\omega = WP_{SEQ}(e_1, e_2)$ as depicted in Figure 6.9.a. If $e_1$ completes and $e_2$ fails afterwards $\omega$ is in an inconsistent (i.e., not semi-atomic) state: $e_1$ cannot be recovered and $e_2$ is not retrievable. $\omega$ is thus not SAP. On the other hand, the workflow $\omega' = WP_{SEQ}(e_1, e_2)$ illustrated in Figure 6.9.b, is SAP: As $e_1$ is retrievable, it cannot fail. However, in case of failure of $e_2$, $e_1$ can be recovered.

![Figure 6.9: Examples of a WP_SEQ which are not (a) [are (b)] correct.](image)

The following proposition states the cases, in which $WP_{SEQ}$ element is correct.$^2$

Proposition 3. SAP of a Sequence $WP_{SEQ}(E)$
A sequence pattern $WP_{SEQ}(E)$ is SAP

\[\iff\]

\(^2\)Recall, that in Section 5.2.1 we defined the elements of $WP_{SEQ}(E)$ to be ordered according to their index.
6.2 Verification of a Workflow

| a. all of its elements $e \in E$ are SAP and |
| b. no transactional conflict \{e_i, e_j\}_C, with $e_i, e_j \in E$ and $i < j$ exists |

Proof.

- ‘$\Rightarrow$’: Assume $WP_{SEQ}(E)$ to be SAP. Claim: All $e \in E$ are SAP and no transactional conflict \{e_i, e_j\}_C, with $e_i, e_j \in E$ and $i < j$ exists. Proof by contradiction:
  - Assume it exists $e \in E$ which is not SAP. Thus, execution of $e$ can result in non semi-atomic termination of $e$ and thus of $WP_{SEQ}(E)$. This contradicts the assumption.
  - Assume it exists \{e_k, e_l\}_C with $e_k, e_l \in E$ and $k < l$. If $e_k (p_T(e_k) = (0, *))$ completes and $e_l (p_T(e_l) = (*, 0))$ fails, the execution is not semi-atomic, thus $WP_{SEQ}(E)$ is not SAP. This contradicts the assumption.

- ‘$\Leftarrow$’: Assume all elements $e \in E$ to be SAP and no transactional conflict \{e_i, e_j\}_C, with $e_i, e_j \in E$ and $i < j$ to exist. Claim: $WP_{SEQ}(E)$ is SAP. Proof by contradiction:
  Assume $WP_{SEQ}(E)$ is not SAP. Then,
  - it either exists an element $e \in E$ which is not SAP and whose execution may cause $WP_{SEQ}(E)$ not to be SAP,
  - or it exists $e_l \in E$ which is not retrieable ($p_T(e_l) = (*, 0)$) and $e_k \in E$ which is not recoverable ($p_T(e_k) = (0, *)$), such that in case of failure of $e_l$ the previous completion of $e_k$ prevents SAP termination. Thus, $l < k$. Then \{e_k, e_l\}_C is transactional conflict.

Both cases contradict the assumption.

\[\square\]

**SAP of $WP_{AND}$ Patterns**

As opposed to sequences, the order of elements within an $WP_{AND}$ is irrelevant.

**Example:** Consider for example, the workflow $\omega = WP_{AND}(e_1, e_2)$ depicted in Figure 6.10.a. If $e_2$ completes while $e_1$ fails, the element is an inconsistent state, as $e_2$ cannot be recovered and $e_1$ is not retrieable. Therefore, $\omega$ is not SAP. On the other hand, the $\omega'$ illustrated in Figure 6.10.b, is SAP: As $e_1$ is retrieable, it will eventually complete (i.e., not fail). However, in case of failure of $e_2$, $e_1$ can be recovered.

The following proposition states the cases in which a $WP_{AND}$ element is correct.
6.2 Verification of a Workflow

![Figure 6.10: Examples of a WP_AND which are not (a) [are (b)] correct.](image)

**Proposition 4.** \( \text{SAP of an AND } WP_{AND}(E) \)

An AND pattern \( WP_{AND}(E) \) is SAP

\[ \iff \]

a. all of its elements \( e \in E \) are SAP and

b. no transactional conflict \( \{e_i, e_j\} \), with \( e_i, e_j \in E \) exists

We employ another observation according SAP execution of \( WP_{AND} \) patterns in order to prove Proposition 4.

**Proposition 5.** \( \text{Transactional Properties of a Conflict Free } WP_{AND}(E) \)

No transactional conflicts \( \{e_i, e_j\} \), with \( e_i, e_j \in E \) in \( WP_{AND}(E) \) exist

\[ \iff \]

a. \( p_T(WP_{AND}(E)) = (1, *) \) or

b. \( p_T(WP_{AND}(E)) = (*, 1) \) or

c. It exists at most one \( e \in E \), with \( p_T(e) = (0, 0) \) and for all other \( e' \in E : p_T(e') = (1, 1) \)

Proof. may be found in Appendix B.1.

Using Proposition 5, we can now easily prove Proposition 4.

Proof.

- \( \implies \): Assume \( WP_{AND}(E) \) to be SAP. Claim: All \( e \in E \) are SAP and no transactional conflict \( \{e_i, e_j\} \), with \( e_i, e_j \in E \) exists. Proof by contradiction:
6.2 Verification of a Workflow

- Assume it exists $e \in E$ which is not SAP. Execution of $e$ might result in non semi-atomic completion of $e$ and thus of $WP_{AND}(E)$. This contradicts the assumption.

- Assume it exists $\{e_k, e_l\} \in C$ with $e_k, e_l \in E$. If $e_k \ (pT(e_k) = (0, *))$ completes while $e_l \ (pT(e_l) = (*, 0))$ fails, the execution is not semi-atomic, thus $WP_{AND}(E)$ is not SAP. This contradicts the assumption.

- ‘$\Leftarrow$’: Assume all elements $e \in E$ are SAP and no transactional conflict $\{e_i, e_j\} \in C$, with $e_i, e_j \in E$ to exist. Claim: $WP_{AND}(E)$ is SAP.

With Proposition 5, we know, that $WP_{AND}(E)$ is then recoverable, redoable or there exists at most one element $e$ with $pT(e) = (0, 0)$ and all other elements $e' \in E$ employ $pT(e) = (1, 1)$. Let us consider these 3 cases:

- If $WP_{AND}(E)$ is recoverable: I.e., all elements $e \in E$ are recoverable, thus failure of any element $e$ is recovered by recovering all other elements $e' \in E$. Thus, $WP_{AND}(E)$ is SAP.

- If $WP_{AND}(E)$ is retrievable: I.e., all elements $e \in E$ are retrievable. Thus, no element can fail, i.e., $WP_{AND}(E)$ is SAP.

- If there exists at most one element $e \in E$ with $pT(e) = (0, 0)$ and all other elements $e' \in E$ employ $pT(e) = (1, 1)$: Since $e$ is the only element in $WP_{AND}(E)$ which is not retrievable, its failure is the only failing situation that might occur. In this case, $WP_{AND}(E)$ is recovered by recovering all $e' \in E \ (e' \neq e)$. As no other failure situation can occur, $WP_{AND}(E)$ is thus SAP.

\[\square\]

In order to preserve semi-atomicity when activating a $WP_{AND}$ pattern, either all of its elements have to be recoverable, all of its elements have to be retrievable, or if there exists at most one element which is not recoverable and not retrievable, all other elements are both: recoverable and retrievable.

**SAP of an XOR**

The execution of a pattern is SAP, if all possible executions of the element result in semi-atomic termination. The definition of the $WP_{XOR}(E)$ pattern infers, that one and only one element $e \in E$ is executed at a time. If an element fails, another element $e' \in E$ is activated. Therefore, the termination of $WP_{XOR}(E)$ implies, that either one and only one element in $e$ completed and all other elements failed or were not activated or all elements failed. Both cases are semi-atomic terminations.
6.2 Verification of a Workflow

Remark 2. **SAP of an XOR WP** \( \text{WP}_{\text{XOR}}(E) \)

An XOR pattern \( \text{WP}_{\text{XOR}}(E) \) is SAP, iff all of its elements \( e \in E \) are SAP.

**SAP of a Subtransaction**

According to its definition, elements enclosed in a subtransaction pattern, are coordinated using an atomic commit protocol. As semi-atomicity is a relaxed notion of atomicity, a \( \text{WP}_{\text{subTA}} \) pattern is always SAP.

Remark 3. **SAP of an Subtransaction WP** \( \text{WP}_{\text{subTA}}(e) \)

A subtransaction pattern \( \text{WP}_{\text{subTA}}(e) \) is always SAP.

### 6.2.3 Verifying a Workflow \( \omega \)

In order to verify a workflow \( \omega \), we regard its tree view \( T_\omega \). The objective is to verify whether the root node in \( T_\omega \) is SAP. For checking if a node \( n \) of \( T_\omega \) is SAP, according to its type, Remark 1 or 2 or Proposition 3 or 4 is employed: Thus, if \( n \) is a leaf, it is SAP; if \( n \) is an inner node, all of its children are recursively verified whether they are SAP. As soon as one node is not SAP, the verification of \( T_\omega \) returns false.

**Example:** Consider for example a workflow \( \omega \) whose tree representation \( T_\omega \) is depicted in Figure 6.11. In order to determine whether the root node is SAP, its children are verified: \( s_1 \) is SAP (Remark 1), however \( \text{WP}_{\text{AND}}(s_2, s_3, s_4) \) is not: Since \( s_3 \) transactionally conflicts with \( s_1 \) and \( s_4 \) (i.e., \( \{s_3, s_2\} \) and \( s_3, s_4 \) exist), the pattern is according to Proposition 4 not SAP. Therefore, the root node is neither SAP (Proposition 3): \( \omega \) is not SAP i.e., the verification for this workflow returns false.

![Figure 6.11: Verification of \( T_\omega \).](image-url)
Analyzing the Runtime of the Verification

Let us briefly review the runtime of the verification. In the worst case (regarding the runtime), $T_\omega$ is completely traversed. The runtime of the algorithm is dependent on the number of (leaf and inner) nodes in $T_\omega$. Thus, it is dependent on the number of services $n$ and the patterns they are being nested in. Without loss of generality, we assume each pattern to enclose at least two elements, i.e., each inner node in $T_\omega$ has at least two children. $T_\omega$ contains the maximal number of inner nodes (worst case), if each workflow pattern contains exactly two elements. I.e., $T_\omega$ is a binary tree, with each inner node having exactly two children. If there are $n$ services in $\omega$, thus $n$ leaves in $T_\omega$ exist, the number of inner nodes is $n - 1$. Therefore, in the worst case, the runtime of the verification algorithm is linear $2n - 1$ in the number of services $n$.

6.3 A-priori Adaptation of the Workflow

If the verification of a workflow fails, it is adapted it in order to ensure correctness. This is done according to the properties of the included elements. Thus, a composite service might be executed as different workflows $\omega$ at different times. In Section 6.5, we introduce how the a-priori algorithms is used during the execution of a workflow.

The adaptation is performed on mandatory workflow elements, thus abstracting from single alternatives in $WP_{XOR}$ patterns. However, if a $WP_{XOR}$ pattern is not SAP (recall Remark 2), the adaptation is performed for each element in $WP_{XOR}$ which is not SAP. For convenience, we refer to the algorithm being executed on a workflow $\omega$ hereafter.

In this section, we at first introduce a theorem about the minimal set of elements which needs to be coordinated, and then specify the algorithm which adapts a workflow in order to guarantee our correctness criterion SAP. In the last part, we demonstrate the correctness of our algorithm in terms of the adaptation and the result.

6.3.1 Minimal Set of Coordinated Elements

In Section 6.2, we proved that the specification of a workflow is correct, if it does not contain certain mannered transactional conflicts. When regarding the nature of transactional conflicts, this implicates that recoverable elements of a workflow have to be aligned prior to non-recoverable elements. Non-recoverable elements can only be followed by retreicable elements.

Following Remark 3, it is easy to conclude, that SAP of a workflow can as well be ensured, if all of its elements are included in a subtransaction. However, as this limits the autonomy of the included elements, we aim at avoiding coordination - if possible. The following proposition claims which mandatory elements have to be coordinated in order to ensure correctness.
6.3 A-priori Adaptation of the Workflow

Theorem 1. **Minimal Set M of Coordinated Mandatory Elements of ω**

Let \( E_{CP} \) be the set of directed transactionally mandatory conflict elements of the workflow \( ω \), i.e.,

\[
E_{CP} := \{ e | \text{e is mandatory in } ω \text{ and } \exists e' \text{ such that } (e, e')_C \text{ or } (e', e)_C \}.
\]

Let further \( E_Z \) be the set of mandatory elements of \( ω \), which are not recoverable and not redoable, i.e.,

\[
E_Z := \{ e | e \text{ is mandatory element in } ω \text{ and } p_T(e) = (0, 0) \}.
\]

\( M \) is defined as follows:

\[
M := \begin{cases} 
\emptyset & \text{if } E_{CP} = \emptyset \text{ and } |E_Z| \leq 1 \\
E_{CP} \cup E_Z & \text{otherwise}
\end{cases}
\]

Then, the following holds:

a. If \( M \) is coordinated using a WP\textsubscript{subTA} pattern i.e., \( WP\textsubscript{subTA}(WP(M)) \), SAP of the workflow can be ensured.

b. If SAP of the workflow is ensured by coordinating a set of elements \( M' \) i.e., \( WP\textsubscript{subTA}(WP(M')) \), then \( M' \) is a superset of \( M \), i.e. \( M' \supseteq M \).

Before we prove this theorem, we would like to note the following: The distinction of cases regarding the definition of \( M \) ensures, that \( M \) is empty, if only one mandatory element \( e \) with \( p_T(e) = (0, 0) \) and no directed transactional conflicts exist. In this case, coordination of this \( e \) is not needed, as SAP is achieved by aligning this element behind the recoverable and before the retrievable elements.

**Proof.**

a. Assume: \( M \) is coordinated. Claim: SAP of the workflow can be ensured. Proof by contradiction:

Assume, there exists a mandatory element \( e_i \notin M \), such that execution of the tuple of elements \( e_i, e_j \) hinders SAP. \( e_i, e_j \) are either aligned in sequence or in parallel. Thus, one of the following is true:

- \( \{e_i, e_j\}_C \in WP\textsubscript{SEQ}(e_i, e_j) \), with \( i < j \). Since \( e_i \notin M \): No data dependency \( e_i \rightarrow e_j \) exists and at most one of them exposes \( p_T(e) = (0, 0) \). Otherwise, \( e_i \) were element of \( M \). Thus, they can be rearranged in \( WP\textsubscript{SEQ}(e_j, e_i) \) which is then SAP. This contradicts the assumption. (Existence of conflict \( \{e_j, e_i\}_C \in WP\textsubscript{SEQ}(e_j, e_i) \), with \( j < i \) results in alignment \( WP\textsubscript{SEQ}(e_j, e_i) \) accordingly.)

- \( \{e_i, e_j\}_C \in WP\textsubscript{AND}(e_i, e_j) \). Since \( e_i \notin M \): No data dependency \( e_i \rightarrow e_j \) exists and at most one of them exposes \( p_T(e) = (0, 0) \). Otherwise, \( e_i \) were element of \( M \). Rearranging the pattern to be \( WP\textsubscript{SEQ}(e_j, e_i) \) (or \( WP\textsubscript{SEQ}(e_i, e_i) \)) ensures SAP. This contradicts the assumption.
b. Assume: SAP of $\omega$ is ensured by coordinating $M'$ (i.e., $WP_{\text{subTA}}(WP(M'))$). Claim: $M'$ is a superset of $M$ (i.e., $M' \supseteq M$). Proof by contradiction: Assume, it $\exists e \in M$, which is not coordinated (i.e., $e \notin M'$). Thus, it holds:

- If $e \in E_{CP} \Rightarrow \exists e' \in M$ with $(e, e')_C$ (or $(e', e)_C$). As a data dependency between these two exists, they have to be aligned in sequence $WP_{\text{SEQ}}(e, e')$ (or $WP_{\text{SEQ}}(e', e)$). According to Proposition 3, this sequence is not SAP. Rearranging is not possible due to the data dependency. Thus, $\omega$ is not SAP. This contradicts the assumption.

- Else if, $e \in E_Z$, then (according to the definition of $M$),

  - another element $e' \in E_Z$ exists. $e$ and $e'$ then form transactional conflicts $\{e, e'\}_C$ and $\{e', e\}_C$. Therefore, neither aligning them in sequence (see Proposition 3), nor in parallel (see Proposition 5) ensures SAP.

  - or a directed transactional conflict $(e_i, e_j)_C \in E_{CP}$ exists (recall $p_T(e_i) = (0, *)$ and $p_T(e_j) = (*, 0)$). $\{e_i, e\}_C$, $\{e, e_j\}_C$ (and $\{e_i, e_j\}_C$) then form transactional conflicts. No sequential alignment of $e, e_i$ and $e_j$ ensures SAP according to Proposition 3, as any alignment regarding the data dependency $e_i \rightarrow e_j$, still contains transactional conflicts. Proposition 5 states, that no parallel alignment of $e, e_i$ and $e_j$ ensures SAP either.

This contradicts the assumption.

Theorem 1 illustrates, that $M$ is the minimal set of mandatory elements which needs to be coordinated to ensure SAP. If this set of elements is enclosed by a $WP_{\text{subTA}}$ pattern, an alignment of all other elements can be found, such that the workflow is correct. In addition, no more elements than $e \in M$ need to be enclosed in a $WP_{\text{subTA}}$ pattern.\(^3\)

### 6.3.2 ATS-Invariant Adaptations

Before we specify, how to adapt the workflow in case verification fails, we comment on the adaptations we pursue. Generally, adaptations of the workflow are altering the execution order, aligning elements in different patterns or eliminating alternatives in $WP_{\text{XOR}}$. However, adaptations of the workflow are only allowed, if they do not alter the semantics of the workflow. This is, the ATS-view of the altered workflow infers the ATS-view of the original:

\(^3\)Indirect conflict elements constitute a solitary exception: Even if included in a $WP_{\text{subTA}}$ pattern, they do not have to be coordinated (see Section 6.2.1.3).
6.3 A-priori Adaptation of the Workflow

Definition 21. ATS-invariant adaptations
An adaptation $f$ altering the control flow of $\omega$ to $\omega'$, formally $f(\omega) = \omega'$ is ATS-invariant, if the ATS-view of $\omega'$, i.e., $ATS_{\omega'}$, implies the ATS-view of $\omega$, $ATS_\omega$:

$$ATS_{\omega'} \Rightarrow ATS_\omega.$$ Therefore, if $ATS_{\omega'}$ is true, $ATS_\omega$ is true as well.

Example: Consider our running example, depicted in Figure 5.3 on page 46. Altering the control flow of the example in the $WP_{AND}$ pattern to be a $WP_{SEQ}(T, R, P)$, i.e. Transportation $T$ before Reservation $R$ before Philharmonics $P$, does not change the ATS-view, as $WP_{SEQ}$ and $WP_{AND}$ expose the same $ATS-status$ (see Section 5.2.1). Eliminating alternative $PCC$, resulting in $\omega'$, is as well an ATS-invariant adaptation, as:

$$ATS_{\omega'} = CRS \land (P \land T \land R) \land Confirm \land (\neg PCC \land PCh)$$

It thus holds $ATS_{\omega'} \Rightarrow ATS_\omega$ (recall $ATS_\omega$, depicted in Figure 6.4 on page 53).

Using ATS-invariance as the correctness criterion of adaptations we are able to show, that our algorithm produces correct results (see Section 6.3.4).

6.3.3 Adaptation Algorithm

Knowing Theorem 1 and the notion of correct adaptations (i.e., ATS-invariant adaptations), we now define our algorithm which adapts a given workflow $\omega$ to a workflow $\omega'$ which ensures SAP. Our algorithm proceeds by traversing the data dependency graph $G_\omega(V, E)$. Since edges represent the existing data dependencies between elements, elements are topologically processed by passing through $G_\omega(V, E)$. While traversing $G_\omega$, we append elements to our output data structure $\omega'$.

If an element $e$ is appended to a pattern $WP(E)$, it is inserted as the last element, thus $WP(E, e)$. By appending an element $e$ to the workflow, we refer to aligning $e$ in sequence to the workflow.

Example: If $e$ is e.g., appended to $\omega = WP_{SEQ}(e_1, e_2)$, the resulting workflow is $\omega = WP_{SEQ}(e_1, e_2, e)$. On the other hand, if $e$ is appended to $\omega = WP_{AND}(e_1, e_2)$, is is aligned in sequence. The resulting workflow is thus: $\omega = WP_{SEQ}(WP_{AND}(e_1, e_2), e)$.

6.3.3.1 Initialization

In order to transform a given workflow $\omega$ into an ATS-invariant workflow $\omega'$, we regard its data dependency graph $G_\omega(V, E)$ and initialize the following sets and variables:

- $V_{CP}$ is the set of all directed transactional conflict elements:

$$V_{CP} := \{e \mid \exists e', \text{ with } e, e' \in V \text{ such that } (e, e')_C \text{ or } (e', e)_C\}$$
6.3 A-priori Adaptation of the Workflow

- \( V_Z \) is the set of all non recoverable and non redoable elements:
  \[ V_Z := \{ e \mid e \in V, \ p_T(e) = (0,0) \} , \]
- \( V_I \) is the set of all indirect conflict elements,
- \( V_M \) is the union of the previous sets as follows:
  \[ V_M := \begin{cases} \emptyset & \text{if } V_{CP} = \emptyset \text{ and } |V_Z| \leq 1 \\ V_{CP} \cup V_Z \cup V_I & \text{otherwise} \end{cases} \]
- \( C \) is the set of all current nodes, i.e., all \( v \in V \) which do not have an incoming edge,
- \( \omega' \) is the output workflow, which is empty at the beginning.

### 6.3.3.2 Avoiding Conflicts – Dealing with XORs

In order to reduce the number of elements which need to be coordinated and thus increase the autonomy of elements, we eliminate alternatives in \( WP_{XOR} \) patterns if thereby transactional conflicts are avoided. As stated in Section 5.2.3, the recoverability of \( WP_{XOR} \) patterns might not be known until runtime. In this case, the properties are either \( p_T(WP_{XOR}(E)) = (?,0) \) or \( p_T(WP_{XOR}(E)) = (?,1) \). Ensuring SAP is possible by assuming the pattern not to be recoverable. However, besides our objective to ensure SAP of a workflow, we additionally aim at minimizing the set of elements which needs to be coordinated, thus included in a \( WP_{subTA} \) pattern.

This is accomplished by eliminating certain elements of \( WP_{XOR} \) patterns to eliminate transactional conflicts. We therefore consider the following cases, in which \( WP_{XOR}(E) \) is part of a conflict and its properties cannot be definitely determined.

![Figure 6.12: Constellation of WPXOR with uncertain properties p_T(XOR) = (?,0).](image)

**a.** If \( p_T(WP_{XOR}(E)) = (?,0) \) (see Figure 6.12), all elements \( e \in E \) either expose \( p_T(e) = (1,0) \) or \( p_T(e) = (0,0) \) and there exists at least one of each kind in \( E \).

---

4 In our examples, we color-code the set \( V_M \) in red.

5 By the ?-symbol, we denote that the property is not known yet.
6.3 A-priori Adaptation of the Workflow

i. If there exists \( e' \in \omega \), such that \( (e', WP_{\text{XOR}}(E))_C \) directly transactionally conflict, deleting any alternatives of \( WP_{\text{XOR}}(E) \) the pattern remains non-retrieable: No element of \( WP_{\text{XOR}}(E) \) is retrieable (recall Definition 25). Thus, the conflict \( (e', WP_{\text{XOR}}(E))_C \) remains. In this case, no branch is eliminated.

ii. If there exists an element \( e' \in \omega \), such that they transactionally conflict \( \{WP_{\text{XOR}}(E), e'\}_C \) or \( (WP_{\text{XOR}}(E), e')_C \), correct execution of them is ensured by deleting all \( e_i \in E \), with \( p_T(e_i) = (0,0) \). The pattern becomes recoverable, \( p_T(WP_{\text{XOR}}(E \setminus \{e_i\})) = (1,0) \) and the conflict is thereby solved. The set \( V_Z, V_{CP} \) and \( V_M \) are updated. This case is illustrated in Figure 6.12 on the right side.

\[ \text{b.i} \] 
\[ \text{b.ii} \]

Figure 6.13: Constellation of \( WP_{\text{XOR}} \) with uncertain properties \( p_T(XOR) = (?,1) \).

b. If \( p_T(WP_{\text{XOR}}(E)) = (?,1) \) (see Figure 6.13), then at least one element \( e \in E \) is redoable. Additionally, at least one element is recoverable and at least one element is non-recoverable. If \( WP_{\text{XOR}}(E) \) is part of a directed transactional conflict (i.e., \( WP_{\text{XOR}}(E) \in V_{CP} \)) with an element \( e' \in \omega \), then the conflict has to be of the following form: \( (WP_{\text{XOR}}(E), e')_C \). According to the existing alternatives, deleting branches of \( WP_{\text{XOR}}(E) \) may alter its properties to be recoverable. However, this may also delete all retrieable elements, resulting in the pattern to be non-retrieable. We therefore distinguish the following cases:

i. If no element \( e \in E \) is recoverable and redoable \( (p_T(e) \neq (1,1), \) see Figure 6.13 on the left side), this conflict may be solved by eliminating all \( e_j \in E \) which are not recoverable \( p_T(e_j) = (0,*) \). Therefore, \( WP_{\text{XOR}}(E') = \)
WP\text{XOR}(E \setminus \{e_j\}) only contains elements which are recoverable however no element, which is redoable anymore (according to the assumption). As WP\text{XOR}(E) was redoable and WP\text{XOR}(E') is not, this might produce a new directed transactional conflict (WP\text{XOR}(E'), e''). If such an e'' ∈ ω exists, no branches of WP\text{XOR}(E) are eliminated. Otherwise, all elements e_j ∈ E which are not recoverable are eliminated.

ii. Else, there exist elements e ∈ E which are recoverable and redoable, thus p_T(e) = (1, 1), see Figure 6.13 on the right hand side. Eliminating all elements which are not recoverable, i.e. all e_j ∈ E, with p_T(e_j) = (0, *) altering the pattern to WP\text{XOR}(E') = WP\text{XOR}(E \setminus \{e_j\}) changes the properties of WP\text{XOR}(E) to be p_T(WP\text{XOR}(E')) = (1, 1). The above conflict is thereby resolved, and WP\text{XOR}(E') is at the most an indirect conflict element. V_CP, potentially V_I and also V_M are updated.

This strategy is assembled as follows. Note, comments reference the enumeration above.

\begin{algorithm}
\textbf{Algorithm 1. Eliminating Branches}
\textit{Input:} w, V_CP, V_Z, V_M

\begin{enumerate}
\item for (all XOR patterns in V_M \ V_I) do {
  \item if (p_T(XOR) = (?, 0)) {
    \item if (XOR not in V_CP && XOR in V_Z) {
      eliminate(all e_i in XOR with p_T(e_i) = (0, 0));
      update V_Z, V_M;
    }
  }
\item else if (p_T(XOR) = (?, 1)) {
  \item if (XOR contains e with p_T(e) = (1,1)) {
    eliminate(all e_j in XOR with p_T(e_j) = (0, *)));
    update V_CP, V_I, V_M;
  }
\item else {
    if (no conflict is produced by elimination) {
      eliminate(all e_j in XOR with p_T(e_j) = (0, *));
      update V_CP, V_M;
    }
  }
}\end{enumerate}

\textit{Output:} V_CP, V_Z, V_M (potentially updated)
\end{algorithm}
Please note, that by eliminating branches, transactional conflicts may be avoided, thereby decreasing the number of elements which need to be coordinated. However, eliminating available alternatives for the sake of autonomy is optional - correctness in the notion of SAP is guaranteed anyhow.

### 6.3.3.3 Recoverable Start

After resolving avoidable conflicts, we start traversing $G_\omega(V,D)$ by processing non-conflicting recoverable elements. Propositions 3 and 4 both state, that a pattern is correct, if no transactional conflicts exist. Therefore, recoverable nodes may be aligned in sequence or parallel without causing transactional conflicts. If more than one current, recoverable node exists, they are aligned in parallel as no data dependency exists among them. Otherwise, they are aligned in sequence.

**Algorithm 2. Adaptation of $\omega$ – Processing Recoverable Elements**

*Input: Data dependency Graph $G_\omega(V,E)$*

```plaintext
while (C contains elements \{r\} with p_T(r) = (1,*)) {
    if (\{|r|\} > 1) {
        append WP_AND(\{r\}) to $\omega'$;
    } else {
        append r to $\omega'$;
    }
    update $G_\omega$, update C;
}
```

This loop is executed until the set of current nodes does not contain any recoverable nodes. Thus, if the set of conflict nodes $V_M$ is not empty, the set of current nodes $C$ now at least contains one element $m$, which is in the set of nodes that need to be coordinated i.e., $m \in V_M$. As only recoverable elements were appended to $\omega'$ and by definition a transactional conflict cannot occur between any two recoverable elements, $\omega'$ is SAP.

### 6.3.3.4 Coordinated Elements

In the next step of the algorithm, elements which need to be coordinated, are processed. As stated above, if $V_M$ is non-empty, there exists at least one element $m \in V_M$ which is also a current node, that is $m \in C$. In Theorem 1, we proved that all elements in $V_M$ need to be coordinated in order to ensure SAP of the workflow.

Therefore, the algorithm continues as follows:

---

\(^6\)This holds, as otherwise there would exist at least one non-recoverable, non-conflicting node $n$ in $C$, from which a path (that is a data dependency) to a non-redoable element $m \in V_M$ existed. As $n$ is non-recoverable it would then transactionally conflict with $m \in V_M$ i.e., $(n,m)_C$. 

---
6.3 A-priori Adaptation of the Workflow

Algorithm. Adaptation of $\omega$ (ctd.) – Coordination of Elements

if $(V_M \neq \{\})$
    $M := \{\}$
    while (C contains elements {m}, with {m} in $V_M$) {
        if ($|\{m\}| > 1$) {
            append $WP_{AND}(\{m\})$ to $M$;
        } else {
            append $m$ to $M$;
        } update $G_w$, update $C$;
    }
    append $WP_{subTA}(M)$ to $w'$;
} else if ($|V_Z| == 1$) {
    append $v$ in $V_Z$ to $w'$;
    update $G_w$, update $C$;
}

Due to the transitivity of conflicts (Proposition 1) and the properties of enclosed conflict elements (Proposition 2), all elements which lie on a path in $G_{\omega}$ from one conflict node $m_i$ to any another $m_j$ are in $V_M$. Either, they are conflicting elements themselves, or indirect conflict elements. Therefore, as soon as the set of current nodes does not contain any $m \in V_M$, $V_M$ is completely processed. Thus, the subtransaction is closed.

In this step, we appended a $WP_{subTA}$ pattern (or only one element $v \in V_Z$) to a recoverable workflow. As $WP_{subTA}$ is SAP, the resulting workflow $\omega'$ is still SAP.

6.3.3.5 Ending Retrievable

By now – if existent – all conflict elements have been processed and appended to $\omega'$. Therefore, only retrievable elements are left to process. If there were a non-retrievable element $v \in C$, $v$ would transactionally conflict with the $WP_{subTA}$ pattern, thus $v$ were a conflicting element and would have been included in $V_M$.

Retrievable elements are appended just as recoverable elements at the beginning.

Algorithm. Adaptation of $\omega$ (ctd.) –Appending Retrievable Elements

while (C contains elements {r}) {
    if ($|\{r\}| > 1$) {
        append $WP_{AND}(\{r\})$ to $w'$;
    } else {
        append $r$ to $w'$;
    }
update G_w, update C;
}
return w';

Output: Workflow $\omega'$, which is SAP, with $ATS_{\omega'} \Rightarrow ATS_{\omega}$.

As only retrieable nodes are left, all remaining elements are appended to $\omega'$ in this step. Just as with recoverable elements, they are arranged in parallel, if no data dependencies exist. Otherwise, they are topologically sorted regarding their dependencies. As no transactional conflicts exist among retrieable elements (see Definition 17), $\omega'$ is SAP. The algorithm terminates, if the set of current nodes is empty: All nodes of $G_{\omega}(V, E)$ have been processed and added to $\omega'$.

6.3.3.6 Example: Adapting a Workflow

In the following, we demonstrate an example. The initial specification of $\omega$ is depicted in Figure 6.14. The verification for $\omega$ fails, as among others transactional conflicts $\{s_3, s_2\}_C$ and $\{s_4, s_2\}_C$ exist. Additionally (among others) directed transactional conflicts $(s_3, s_5)_C$, $(s_4, s_5)_C$ and $(s_5, s_6)_C$ exist. Furthermore, $\{s_{x5}, s_{x4}\}_C$ transactionally conflict, thus $X_2 = WP_{XOR}(WP_{AND}(s_{x4}, s_{x5}), s_{x6})$ cannot be verified.

Therefore, in the first run, $WP_{AND}(s_{x4}, s_{x5}) \in X_2$ is adapted as follows: As no data dependency exists among $s_{x4}, s_{x5}$, $G_X(V, E)$ consists of two vertices ($s_{x4}$ and $s_{x5}$) and no edge. Therefore, the recoverable element $s_{x4}$ is aligned before the non-recoverable, non-redoable element $s_{x5}$. Thus, the resulting $WP_{XOR}$ pattern is:

$$X_2' = WP_{XOR}(WP_{SEQ}(s_{x4}, s_{x5}), s_{x6})$$

In the next run, the whole workflow $\omega$ is adapted. Its data dependency graph $G_{\omega}(V, D)$ is depicted in Figure 6.15.
6.3 A-priori Adaptation of the Workflow

Initialization The set of conflicting elements $V_M$ is initialized as $V_M = \{s_3, s_4, s_5, X_1, s_6\}$. These are color-coded in red in Figure 6.15. The set of current nodes $C$ consists of one single node $C = \{s_1\}$. The output workflow $\omega'$ is initialized as well.

Eliminating Branches In this step, elements of $WP_{XOR}$ patterns which need to be co-ordinated are eliminated, if this reduces the number of conflicts. In our example, $X_1$ is the only $WP_{XOR}$ pattern which is $\in V_M$. It is retrievable, however its recoverability is not known, i.e., $p_T(X_1) = (?, 1)$. It therefore potentially conflicts with e.g., $s_5$ as well as $s_6$. As $s_{x3}$ with $p_T(s_{x3}) = (1, 1)$ is part of the pattern, by eliminating all elements in $X_1$ which are not recoverable (cf., case b.ii in Section 6.3.3.2) are eliminated. This results in $X'_1 = WP_{XOR}(s_{x2}, s_{x3})$. $X'_1$ is now both recoverable and retrievable, thus does not conflict with other nodes anymore. However, it is now an indirect conflict element. Therefore, it is color-coded in light red in the following.

Processing Recoverable Elements We start traversing the data dependency graph by appending recoverable elements to $\omega'$. As $s_1$ is the only current node, it is aligned at the beginning. The according nodes and edges are deleted from the graph (Figure 6.16). The set of current nodes now contains $C = \{s_2, s_3, s_4, s_7\}$. $s_2$ and $s_7$ are both recoverable. Thus, $WP_{AND}(s_2, s_7)$ is appended to $\omega'$ (Figure 6.19). After refreshing $G_\omega(V, D)$ and
6.3 A-priori Adaptation of the Workflow

$C = (s_3, s_4)$ once more, the set of current nodes does not contain recoverable elements anymore (see Figure 6.17). Therefore, the algorithm proceeds with the next step.

Figure 6.17: $D_\omega(V, E)$ of the example - all recoverable elements processed.

**Coordination of Elements** All conflicting elements are appended within a subtransaction in this step. Therefore at first, $s_3$ and $s_4$ are appended in parallel. They are succeeded by $s_5$ and by the indirect conflict element $X_1'$. The last element, which needs to be coordinated is $s_6$. Thus, in this step,

$$WP_{subTA}(WP_{SEQ}(WP_{AND}(s_3, s_4), s_5, WP_{XOR}(s_{x2}, s_{x3}), s_6))$$

is appended to $\omega'$, as it can be seen in Figure 6.19. As stated before $X_1'$ is an indirect conflict element, i.e., it is included in the $WP_{subTA}$ pattern. However, it does not need to be coordinated. If $s_5$ completes, $s_{x2}$ is invoked. In case it fails, $s_{x3}$ is activated as an (retrieable) alternative. If after the execution of $X_1'$, $s_6$ votes, that it cannot successfully commit, $X_1'$ can be recovered (as both of its elements are recoverable).

Figure 6.18: $D_\omega(V, E)$ of the example - coordination of elements finished.

**Appending Retrievable Elements** By now, the set of current elements $C$ only contains retrievable elements ($C = \{X_2', s_8\}$). Therefore, these are appended to $\omega$ in parallel, $WP_{AND}(WP_{XOR}(WP_{SEQ}(s_{x4}, s_{x5}), S_{x6}), s_8)$. Thereby all nodes in $G_\omega(V, E)$ have been processed and the algorithm terminates. The output is depicted in Figure 6.19.
### Analyzing the Runtime of the Adaptation Algorithm

Let $\omega$ be the input to the algorithm, and $G_\omega(V, D)$ the according data dependency graph. Let $n_v$ denote the number of nodes in $G_\omega(V, D)$ ($n_v = |V|$) and $n_x$ the number of $WP_{XOR}$ patterns in $G_\omega(V, D)$. During the *initialization*, all conflicting elements are identified. In a brute-force approach to identify all these elements, all $v \in V$ are traversed and for each of them it is determined whether $v \in V_Z$ or $e \in V_{CP}$. This may be done in $\leq v_n^2$ steps.\(^7\) For the elimination of branches, each $WP_{XOR}$ node with undetermined properties $v_x \in V$ which is $\in V_Z \cup V_{CP}$ is regarded. For each such $WP_{XOR}$ node, its transactional conflicts are reviewed. Again, performing a brute-force approach, $n_x * |V_Z \cup V_{CP}|$ steps are needed. As $n_x \leq n_v^8$ and $|V_Z \cup V_{CP}| \leq n$: $n_x * |V_Z \cup V_{CP}| \leq v_n^2$. In the remaining steps of the algorithm, $G_\omega(V, D)$ is traversed exactly once. Thus, the remainder of the algorithm requires $n_v$ steps. Overall, the runtime of the (brute force approach of the) algorithm is thus $\leq 2 * n_v^2 + n_v$, thus quadratic in the number of nodes $n_v$ in $G_\omega(V, D)$.

### 6.3.4 Correctness of the Algorithm

In order to demonstrate the correctness of the adaptation algorithm, we show that for every workflow $\omega$, the algorithm produces an output workflow $\omega'$, which is ATS-invariant to $\omega$ and which is also correct i.e., SAP. Additionally, we show that the algorithm is optimal in terms of the number of coordinated elements.

---

\(^7\)In practice, less steps are needed as for each non-recoverable node $v$, only nodes $v'$ to which a path from $v$ to $v'$ in $G_\omega(V, D)$ exists, are regarded. Additionally, the transitivity of conflicts can be beneficially exploited to further reduce the number of steps.

\(^8\)presumably $v_x \ll n_v$
6.3 A-priori Adaptation of the Workflow

6.3.4.1 ATS-Invariance of Resulting $\omega'$

Correctness of the Algorithms 1 and 2 includes correctness of the adaptations: i.e., the ATS of the adapted workflow $\omega'$ are invariant those of the input workflow $\omega$. By comparing the ATS-view of the original workflow $\omega$ with the ATS-view of the resulting workflow $\omega'$, we are able to demonstrate the ATS-invariance of our adaptations. The proposed algorithms may eliminate alternatives in $WP_{XOR}$ patterns and re-arrange elements of $\omega$. It is therefore straightforward however tedious to show that for the output workflow it holds: $ATS_{\omega'} \Rightarrow ATS_{\omega}$. The interested reader is referred to the Appendix B.2.

6.3.4.2 SAP of $\omega'$

In this section, we show, that the output of the Algorithm 2 is correct in the notion of $SAP$. That is, if it produces correct outputs $\omega'$ for allowed inputs. We demonstrate, that the resulting workflow $\omega'$ is $SAP$. We proceed as follows: We define the invariant $\omega'$ is $SAP$ and demonstrate that it holds in every step of the algorithm.

**Initialization** During this step, $\omega'$ is initialized void. As an empty workflow cannot be executed, it is $SAP$ by definition.$^9$ It therefore holds, that $\omega'$ is $SAP$.

**Eliminating Branches** During this step of the algorithm, $\omega'$ is not altered. Therefore, $\omega'$ is still correct i.e., it is $SAP$.

**Processing Recoverable Elements** In this step, the algorithm processes recoverable elements, which are not part of directed transactional conflicts. These are appended to $\omega'$. This is repeated until the set of current nodes does not contain any recoverable non-conflicting nodes $\{r\}$. If no recoverable element exists, the invariant holds, $\omega'$ is $SAP$ after this step (as it is still void).

Otherwise, let $\omega'_i$ denote $\omega'$ after the $i$th iteration. $\omega'_0$ is thus the state of $\omega'$ before any recoverable elements are appended i.e., $\omega'_0$ is void. Depending on the number of recoverable elements in the set of current nodes $\omega'_i$ is determined by:

$$
\omega'_1 = \begin{cases} 
WP_{AND}(\{r\}), & \text{if } |\{r\}| > 1 \\
r, & \text{if } |\{r\}| = 1 
\end{cases}
$$

In any case, $\omega'_1$ is $SAP$, as it either consists of one recoverable service, or a $WP_{AND}$ pattern, which is recoverable and thus $SAP$ according to Proposition 5. For any further iteration ($i > 1$) in this step of the algorithm it holds:

$^9$Its execution cannot result in non semi-atomic termination.
6.3 A-priori Adaptation of the Workflow

\[ \omega' = \begin{cases} 
WSEQ(\omega'_{i-1}, WPAND(\{r\})) & , \text{if } |\{r\}| > 1 \\
WSEQ(\omega'_{i-1}, r) & , \text{if } |\{r\}| = 1 
\end{cases} \]

As \( \omega'_{i-1} \) is recoverable (and certainly \( r \) and \( WPAND(\{r\}) \) are recoverable), it holds (according to Proposition 3):

1. \( p_T(\omega'_i) = (1, ?) \)
2. \( \omega'_i \) is SAP

Therefore the invariant holds for every iteration and thus also holds after the last iteration i.e., this algorithm step.

**Coordination of Elements** Elements, which need to be coordinated are appended in this step of the algorithm. As just demonstrated, the output \( \omega' \) of the previous step is SAP. Let \( \omega'_{j0} \) denote the input of this step of the algorithm. \( \omega'_j \) is then determined in the following way:

\[ \omega'_j = \begin{cases} 
WSEQ(\omega'_{j0}, WP_{subTA}(WP(V_M)))) & , \text{if } V_M \neq \emptyset \\
WSEQ(\omega'_{j0}, v_z) & , \text{else if } V_M = \emptyset \text{ and } |V_Z| = 1 \\
\omega'_{j0} & , \text{otherwise} 
\end{cases} \]

Regarding the definition of the \( WP_{subTA} \) pattern, the pattern is always SAP. Just as \( v_z \) it holds the properties \( p_T(v_z) = p_T(WP_{subTA}) = (0, 0) \). If \( \omega'_{j0} \), which is the output of the previous algorithm step, is empty, \( \omega'_j \) is obviously SAP. Otherwise, \( \omega'_{j0} \) is recoverable. As to Proposition 3, aligning a recoverable element prior to a non-recoverable element is SAP. Therefore, the invariant is true. If elements are appended in this step (i.e., \( V_M \cup V_Z \neq \emptyset \)), the properties of \( \omega'_j \) are \( p_T(\omega'_j) = (0, 0) \).

**Appending Recoverable Elements** In the last step, the algorithm solely processes retrievable elements, which we denote by \( \{r\} \). These are appended to \( \omega' \). If no retrievable elements exist, that is \( \{r\} = \emptyset \), \( \omega' \) is not modified and thus remains SAP.

Otherwise, let \( \omega'_k \) denote \( \omega' \) after the \( k \)th iteration of this step. \( \omega'_{k0} \) is thus the state of \( \omega' \) before any retrievable elements is appended. \( \omega'_{k0} \) is SAP. Depending on the number of retrievable elements in the set of current nodes, for \( k > k_0 \), \( \omega'_k \) is determined by:

\[ \omega'_k = \begin{cases} 
WSEQ(\omega'_{k-1}, WPAND(\{r\})) & , \text{if } |\{r\}| > 1 \\
WSEQ(\omega'_{k-1}, r) & , \text{if } |\{r\}| = 1 
\end{cases} \]

Certainly, \( WPAND(\{r\}) \) and \( r \) are retrievable. Following Proposition 5, \( WPAND(\{r\}) \) is SAP. Aligning a retrievable element (which is SAP) behind any element (which is also SAP) results as per Proposition 3 in a correct sequence. Therefore, the invariant holds for every iteration of this step and especially is inherent after the algorithm terminates. Thus, the algorithm produces correct outputs in the notion of SAP.
6.3.4.3 Minimality of Coordinated Elements

We previously demonstrated that the adaptation algorithm produces correct output workflows $\omega'$, which are ATS-invariant to the according input. We now argue, that the adaptation algorithm is optimal in terms of the number of coordinated elements.

Theorem 1 states the minimal set of elements which has to be coordinated in order to guarantee $SAP$ of the workflow. Thus, by showing that the adaptation algorithm coordinates exactly these elements, we are able to state, that our algorithm produces the optimal result regarding the number of coordinated elements.

The definition of $M$ matches the definition of $V_M$ in the algorithm, besides the indirect conflict elements $V_I$. Thus, $M = V_M \setminus V_I$. Following the definition of indirect conflict elements (Definition 19), as these expose full flexibility, they do not have to be coordinated in any case.

As stated before, we forego an explicit notation for indirect conflict elements. They are identified as the elements $e$ within a $WP_{subTA}(e)$ pattern with the transactional properties $p_T(e) = (1, 1)$. However, we also previously stated, that – although included in a $WP_{subTA}$ pattern – they are not coordinated.

Thus, the set of elements, which is coordinated in $\omega'$ (that is $V_M \setminus V_I$) corresponds to the minimal set of coordinated elements $M$. Therefore, our algorithm produces the optimal result regarding the number of coordinated elements.

6.4 Integrating Service Discovery in Adaptive Workflow Management

In Chapter 4 we introduced a protocol to discover mobile services. We previously argued, that service discovery is essential to be able to find cooperating entities, especially in dynamic environments, e.g. mobile networks. However, service discovery is also significant for the proposed flexible workflow management in the following way: By employing service discovery, alternatives of services may be found and integrated as elements of $WP_{XOR}$ patterns in the workflow. The advantages of this approach are the following.

By integrating alternatives, the transactional properties of an $WP_{XOR}$ pattern are altered and additionally flexibility might be obtained. I.e., as introduced in Section 5.2.3, an $WP_{XOR}$ pattern becomes retrievable if one retrievable alternative is integrated. Thus, by integrating alternatives, transactional conflicts are likely to be avoided.

On the other hand, as introduced in second step of the algorithm (see Section 6.3.3.2), alternatives may be eliminated in order to solve transactional conflicts. Thereby, less coordination is needed which yields to increased autonomy of the involved services. This also ensures, that no additional conflicts arise due to the integration of alternatives: Conflicting alternatives are expunged from the workflow.

Last but not least, integration of service discovery in flexible workflow ensures forward-
recovery during execution. If the invocation of services continuously fails (e.g., they are not available anymore), new services are integrated to still enable successful completion.

All in all, the combination of service discovery and adaptive workflow management, as proposed in this thesis, is essential for ad-hoc cooperation: In the first place, as dynamically services can only be bound if discovered. On the other hand, if several providers are discovered, forward-recovery through the integration of alternatives in $WP_{XOR}$ patterns is enabled. This increases the overall chance of such workflows to be successfully completed as well as the autonomy of participants (see Section 8.3.2).

### 6.5 Adaptation at Runtime

So far, using our formal model (i.e., the properties of elements) we abstracted from the mobility of single components. As stated in the previous section, services may be dynamically bound during execution: E.g., if a service is not available anymore and needs to be replaced. So far, we introduced our a-priori adaptation algorithm which adapts a workflow $\omega$ prior to execution. However, it is also be applied during the execution of $\omega$, if failures of elements repeatedly occur.

Recall our formal model: We assume the compensation of a compensatable service to be available (e.g., via reliable communication channels). Further, we assume retrievable services to as well ensure availability via reliable communication channels. Due to these assumptions, a processed workflow is at any time during the execution guaranteed to be SAP, i.e., either recoverable ensuring semi-atomicity or retrievable thus guaranteeing successful completion. This is given, as either the verification of $\omega$ outputs that $\omega$ (i.e., the root note of $T_\omega$) is SAP or the adaptation algorithm converts $\omega$ to be SAP.

If an element $e$ continuously fails at runtime and needs to be replaced, it cannot be retrievable (i.e., $p_T(e) = (\ast, 0)$). The execution of $\omega$ up to $e$ is recoverable. The execution of $\omega$ is then interrupted and employing service discovery, alternatives are searched for. If no alternatives are found, $\omega$ is backward-recovered, preserving semi-atomicity. On the other hand, if appropriate alternatives $e'$ for $e$ are discovered, we distinguish the following cases:

1. If $p_T(e) = p_T(e')$, $e$ is simply replaced by $e'$ and the execution of $\omega$ is continued ensuring SAP.

2. Else, if $e$ was element of a $WP_{subTA}$ pattern and
   
   a. $e$ is non-recoverable ($p_T(e) = (0, 0)$) and $e'$ is recoverable ($p_T(e) = (1, \ast)$) or
   
   b. $e'$ retrievable (i.e., $p_T(e') = (\ast, 1)$)

   coordination of $e'$ might become obsolete.
3. On the other hand, if $e$ was not element of a $W_{\text{sub} TA}$ pattern and $e'$ is not recoverable (i.e., $p_T(e') = (0, *)$), re-arranging $\omega$ or coordination of $e'$ and following elements might become necessary to preserve SAP.

In cases 2 and 3, re-execution of the adaptation algorithm, as described in Section 6.3, is necessary to ensure SAP or respectively to avoid coordination if possible. In these cases, the adaptation is executed on the remaining nodes of $G_{\omega}(V, D)$ which have not been executed yet.

Thereby, integrating service discovery to our approach of flexibly altering workflows \textit{during} execution enables forward-recovery in case of failure of services. It is thereby very well suitable to cope with mobile services, whose availability might vary.
7 Implementing Adaptive Workflow Management

In this chapter, we present the implementation of the adaptive workflow management system (AWM for short). It assembles the formal model as introduced in Chapter 5 and the algorithms to verify and adapt workflows to ensure semi-atomicity (Chapter 6). As stated in the introduction, we base our implementation on Web Services. We are thereby able to make recourse to a broad range of existing specifications, especially WS-Tx (Section 3.1.1) and BPEL (Section 3.1.2). The implemented system is based on the BPEL engine Apache ODE. The illustrations of AWM thus rely on ODE specific BPEL elements.

The presentation of the implementation is divided into the following parts: At first, we illustrate the realization of the formal model. We present the architecture of the adaptive workflow management system and conclude by introducing its use cases.

7.1 Formal Requirements

In order to be able to implement the algorithms to ensure semi-atomicity as introduced in Chapter 6, the formal model as specified in Chapter 5 has to be realized. Therefore, we present the implementation of the transactional properties, the workflow patterns and the integration of the transactional composition employing the failure handling mechanisms offered by Apache ODE.

7.1.1 Transactional Properties of Services

AWM explores transactional service properties to ensure correct service composition. Thus, on the one hand, Web Services have to be labeled according to their transactional properties. This is done using the WS-Policy specification\(^1\). On the other hand, these properties have to be properly integrated as well. That is, if for example, a service is compensatable, the according failure handling is automatically added. Additionally, if a service is redoable, AWM ensures its repeated invocation in case of failure.

\(^1\text{http://www.w3.org/Submission/WS-Policy/}\)
7.1 Formal Requirements

7.1.1.1 Modeling as WS-Policies

The annotation of WS according to their complete transactional properties is done using WS-Policy. WS-Policy is a WS-* specification that enables service providers to define machine-readable directives for their usage. Thereby, a provider is able to specify, e.g., guidelines regarding required security mechanisms or quality of service tags. These directives are integrated in a service’s WSDL description.

AWM provides an XML-schema which specifies the annotation of transactional properties of services. In Listing 7.1, the annotation for a service $s$ with transactional properties $p_{CT}(s) = (0, 1, 0)$ is depicted.

```xml
<wsp:Policy trans:id="tp010">
  <wsp:ExactlyOnce>
    <wsp:All>
      <trans:transactional>
        <trans:compensatable>false</trans:compensatable>
        <trans:consistentCompletion>true</trans:consistentCompletion>
        <trans:retrieable>false</trans:retrieable>
      </trans:transactional>
    </wsp:All>
  </wsp:ExactlyOnce>
</wsp:Policy>
```

Listing 7.1: WS-Policy tp010

AWM allows for these policies to be bound to the <port> element of a service in order for each WS to define its respective properties. An example binding for policy tp010 to the TransportationService is shown in Listing 7.2.

```xml
<service name="TransportationService">
  <port name="TransportationServicePort"
    binding="tns:TransportationServicePortBinding">
    <wsp:PolicyReference URI="#tp010" />
    <soap:address location="http://localhost:8888/Transportation" />
  </port>
</service>
```

Listing 7.2: Binding of WS-Policy tp010 to TransportationService

If a WS does not specify a transactional policy, its default properties are assumed to be as specified in Listing 7.1. Thus, it is neither assumed to be recoverable, nor retrievable. Thereby, the correctness of the algorithm is nevertheless ensured.

7.1.1.2 Integration in Workflows

According to the specified transactional properties, AWM automatically realizes the appropriate handling for a service. That is, if a service is compensatable, the invocation of the compensation service is integrated in the workflow using Apache ODE’s compensation handlers (cf. Section 7.1.3).
Additionally, if a service is retrievable, it assures that it will eventually completes, if its activation is repeated in case of failure. Therefore, AWM integrates the service in the BPEL process as depicted in Figure 7.1. The invocation is repeated if the services fails (as specified in Section 5.1.1).

Figure 7.1: Integration of a retrievable service.

7.1.2 Control Flow Patterns

In Chapter 5, we identified four patterns to be significant for transactional support of workflows (\(WP_{SEQ}\), \(WP_{AND}\), \(WP_{XOR}\) and \(WP_{subTA}\)). These patterns are implemented in BPEL as follows:

The \(WP_{SEQ}\) pattern directly corresponds to the BPEL element \(<sequence>\). The \(WP_{AND}\) pattern likewise complies with the BPEL element \(<flow>\).

The \(WP_{XOR}\) pattern is implemented using case statements \(<if>\). The elements of an \(WP_{XOR}\) pattern are sequentially arranged in the way, that in case of failure of an element, the next one is invoked as an alternative. That is done by defining empty failure handlers and setting appropriate internal variables (see Appendix C). Thereby, AWM realizes the specified alternative dependencies (cf. Section 5.2.5). In order to distinguish these alternative constructs from conventional if-then-else constructs, the \(WP_{XOR}\) patterns are enclosed in a separate BPEL scope. Per naming convention, these are identified by the prefix “\(XOR\)”.
The $WP_{subTA}$ pattern is also encapsulated in a distinct scope, which is likewise labeled using the prefix “SubTA.” Apache ODE supports WS-AT using atomic scopes\(^2\), i.e., `<scope aomitc="yes">`. These employ the use of distributed commit protocols as introduced in Section 3.1.1.1 to coordinate the enclosed services. Thus, the requirements for the $WP_{subTA}$ pattern as specified in Section 5.2.2 are met.

### 7.1.3 Transactional Composition of Services

The failure handling specified in Section 5.2.5.3 to ensure semi-atomicity of a workflow, is realized using the compensation and failure handlers provided by BPEL.

If a Web Service is specified to be compensatable using the WS-Policy as outlined in the previous section, AWM automatically adds the appropriate compensation handler to the invocation of that service. The handler itself invokes the compensation service associated to this Web Service. In Listing 7.3, a compensation handler is added to the invocation of the TransportationService. It invokes the callCompensation method in case of failure of the call method. The compensation handlers of BPEL are used by AWM to implement the appropriate dep\(*Cps\)-dependencies (see Section 5.2.5).

```xml
<bpel:invoke name="TransportationService"
  partnerLink="TransportationServiceLink"
  operation="call"
  portType="tsp:TransportationServicePortType"
  inputVariable="TransportationServiceInput"
  outputVariable="TransportationServiceOutput">
  <bipel:compensationHandler>
    <bipel:invoke name="Compensate_TransportationService"
      partnerLink="TransportationServiceLink"
      operation="callCompensation"
      portType="tsp:TransportationServiceCompensationPortType"
      inputVariable="TransportationServiceInput"
      outputVariable="TransportationServiceOutput">
    </bipel:invoke>
  </bipel:compensationHandler>
</bipel:invoke>
```

Listing 7.3: Specification of the according compensation handler.

In case of failure during the execution of a BPEL process, either explicitly thrown (i.e., using a `<throw>` element), failure of an invoked service, or internal errors (e.g., variable type mismatch), the failure handling of BPEL is launched. Using explicit `<faultHandlers>`, AWM invokes the defined compensation handlers of previously completed services by employing `<compensation>` elements.

Additionally, by calling BPEL’s default-failure handling, all active elements are ensured to be canceled if necessary. Thereby, AWM realizes the dep\(*Chn\)-dependencies as specified in the formal model.

\(^2\)http://ode.apache.org/atomic-scopes-extension-for-bpel.html
7.2 Architecture

Using the encapsulation of elements in $WP_{XOR}$ patterns as well as the described compensation and failure handling, AWM realizes all specified failure recovery dependencies.

7.2 Architecture

In this section, we present the components of AWM in detail and illustrate their interaction. The architecture of AWM is depicted in Figure 7.2. AWM is implemented as a web application which runs within an web server, i.e., in our case Apache Tomcat$^3$. As it can be seen in Figure 7.2, AWM’s architecture complies with the classical layers of a web application: The presentation layer is - as its name indicates - responsible for the presentation of contents and interaction with the user. The logical layer encapsulates AWM’s business logic. In the data layer, internal data is persisted in a relational database, in our case MySQL$^4$. As already mentioned, AWM employs Apache ODE to execute the deployed BPEL processes. We introduce the components of AWM’s presentation and logical layer more closely in the following.$^5$

Figure 7.2: Architecture of AWM.

7.2.1 Presentation Layer

The presentation layer implements the user interface. Thereby, the user is able to access the implemented business logic. The presentation layer is notified of the application’s state by the logical layer and displays the according views. It implements the use cases, which are illustrated in the next section (Section 7.3).

The presentation layer is implemented using Stripes$^6$. Stripes is a web framework which employs Java technologies to enable rapid presentation layer development. As it

$^3$http://tomcat.apache.org/
$^4$http://www.mysql.com/
$^5$As the functionality of the data layer is straightforward, we forego further explanation.
$^6$http://www.stripesframework.org
7.2 Architecture

is a lightweight framework and aims at keeping the configuration overhead low, we chose Stripes as the presentation framework for AWM.

7.2.2 Logical Layer

The logical layer of AWM is divided into three main components: The AWMServices which implement the business logic, the Store and the DAO (Data Access Object).

AWMServices  The AWMServices are not to be confused with the Web Services which are element of BPEL processes. The AWMServices component supplies the algorithms present to the business logic of AWM. On the one hand, it provides service discovery and service integration to integrate dynamic services into uploaded processes. Depending on the underlying networking infrastructure, it is desirable to integrate diverse discovery mechanisms. If more than one suitable service is discovered, AWM enables forward recovery through integrating these as alternatives in \( WP_{XOR} \) patterns. On the other hand, AWMServices incorporate functionality to verify and adapt uploaded processes.

Store  The Store is the central component of the logical layer. It operates the control flow and communicates with other components. It receives user requests, e.g., uploaded BPEL processes or requests to execute previously deployed processes (reception).

If the Store receives an uploaded archive, it extracts all necessary information and passes it to the database. As the AWMServices utilize an object model of the BPEL process, the Store transforms the BPEL description of the process into the internal object representation (un-marshalling). Accordingly, the Store takes verified and adapted object representations of processes and in turn generates valid BPEL descriptions (marshalling). For handling purposes, including marshalling and un-marshalling, AWM utilizes JAX-B\(^7\) and JAX-WS\(^8\).

The BPEL representation of the verified process along with all necessary information (e.g., deployment descriptor) is packaged by the Store and deployed to the BPEL Engine. The information of the process (e.g., its URL) are inserted into the database. If a process is invoked by the user, the Store is responsible for mapping the given URL to the according internal address.

DAO  The DAO represents the interface to the data stored in the database. The DAO is implemented using the Hibernate persistence framework\(^9\) to maintain internal data. Thereby, processes may be stored to and loaded from the database.

\(^7\)https://jaxb.dev.java.net/
\(^8\)https://jax-ws.dev.java.net/
\(^9\)https://www.hibernate.org/
7.3 Use Cases

As already mentioned, there are two main use cases of AWM: On the one hand, users are able to deploy processes, on the other hand deployed processes can be invoked.\(^{10}\)

7.3.1 Deployment

Deployment of a process to a BPEL engine implies the compilation of the process and returning a URL under which the it may be invoked. Apache ODE expects a WAR-archive, which aside from the BPEL description of the process consists of the WSDL description of the process and the included Web Services, the XML-schema of the utilized data structures and the deployment descriptor of the BPEL process.

If a user uploads such an archive, the Store receives the archive, unpacks it, stores all necessary information, generates the object representation (un-marshalling), and forwards this representation to AWMServices. If all needed services are discovered and integrated, the process is verified and adapted (if the verification fails). The Store transforms the verified (and respectively adapted) process to a valid BPEL representation (marshalling). It packs it along with all necessary information (WSDL files, XML schema, deployment descriptor) and deploys the process to the BPEL engine.

The address of the process returned by Apache ODE is managed by AWM and is along with all necessary information stored in the database. If the deployment is successful, the confirmation is returned to the user, as it can be seen in Figure 7.3.

---

\(^{10}\)For a more detailed presentation of the use cases, we refer to [H09].
7.3.2 Process Invocation

The invocation of a process is encoded in a SOAP message and bound to the URL under which the process is deployed (see address information in Figure 7.3). The Store receives the request from the frontend. According to the information in the database, it is able to identify the corresponding process. AWM checks the availability of the dynamically bound services and integrates potentially discovered alternatives.

If new services are integrated into the process, it is again validated, if necessary adapted and re-deployed. This is done just as in the according steps of the deployment. The Store forwards the request to the URL, under which the validated process is deployed. Thereby, the validated process is executed in the BPEL engine. Finally, AWM forwards the result of the invocation returned by Apache ODE to the user. As Apache ODE does not support alterations of BPEL processes during execution, AWM by now implements verification and adaptation prior to execution.
8 Evaluating AWM

In this chapter, we evaluate the adaptive workflow management which implements the model and algorithms introduced in Chapter 5 and 6.\footnote{Results have partly been published in [HS09b, Hah10].} We present relevant system parameters and employed evaluation metrics in Section 8.1. In Section 8.2, we provide an empirical evaluation of our approach, the adaptive workflow management system (AWM), in a variety of settings. Results are classified according to the used metrics. At first, the degree of autonomy is evaluated. We thereby compare our approach to a pessimistic approach of transactional workflow management, i.e. WS-AT (cf. Section 3.1.1.1). After that, we focus on results regarding the semi-atomicity probability of a specified workflow. Using these, we demonstrate the benefits of AWM as opposed to an optimistic approach to transactional workflow management WS-BA (cf. Section 3.1.1.2).

In the last part of this chapter (8.3), we study the performance for AWM in realistic exemplary settings. We thereby present the results for the integration of service discovery into adaptive workflow management system (Section 8.3.2.2): We demonstrate the influence of discovery of alternatives as described in Section 6.4.

8.1 System Parameters and Evaluation Metrics

Prior to presenting the metrics used to quantify the benefits of AWM, an introduction the relevant system parameters that are varied in our experiments is given.

Influential System Parameters

Since the following parameters influence the behavior (and thus the results) of the considered approaches, we vary these in our series of tests.

Number of Included Elements The size of the workflow is denoted by the number of included elements $n$.

Ratios of Elements with Transactional Properties The transactional properties $p_T(s)$ of a service $s$ determine whether $s$ transactionally conflicts with other elements. Therefore, we vary the ratio of services in a workflow $\omega$ which expose certain transactional properties.
8.1 System Parameters and Evaluation Metrics

- $p_{RC}(\omega)$ denotes the ratio of recoverable services in $\omega$, i.e., $p_{RC}(\omega) \times n$ elements in $\omega$ are recoverable.\(^2\)
- $p_{RD}(\omega)$ denotes the ratio of redoable services in $\omega$. That is, $p_{RD}(\omega) \times n$ elements are redoable.\(^3\)

**Data Dependencies** On the one hand, we vary the *number* $t$ of data dependencies, which exist within a given workflow. On the other hand, we vary the *length* $l$ of data dependencies ($s_{i1} \rightarrow \ldots \rightarrow s_{il}$), that is the number of elements in a given data dependency sequence (see Figure 8.1). The overall *ratio* of elements, which are involved in data dependencies is denoted by $r$ (e.g., $r = \frac{t \times l}{n}$).

![Figure 8.1: Number $t$ and length $l$ of data dependencies.](image)

**Success probability** The success probability $p_S(s_i)$ of a single service $s_i$ denotes the probability that this service successfully completes. Thus, the chance that $s_i$ fails is $1 - p_S(s_i) = \overline{p_S(s_i)}$. Unless stated otherwise, we assume homogeneous environments, thus success probabilities of all services in a workflow are normally distributed around a stated mean (denoted as $p_S$) and a variance of 5%.

**Workflow Patterns** The initial alignment of elements in workflow patterns does not influence the output of our algorithm (Section 6.3.3). However, it influences the probability that a given workflow $\omega$ semi-atomically terminates employing WS-BA. Thus, we consider the *parallel* and *sequential* alignments of elements in the according series of tests.

In the following, we introduce the metrics that are employed to quantify the performance of AWM. Furthermore, we present an analytical approach to approximate the results of our experiments.

\(^2\)As an element is defined to be recoverable if it is compensatable or does not need consistent closure, $p_{RC}(\omega)$ is determined by inspecting the ratios of the elements which are compensatable respectively demand consistent closure. As no additional insights are obtained by considering the ratios of compensatable services and those which demand consistent closure separately, we solely present results in which $p_{RC}(\omega)$ is varied.

\(^3\)For reasons of clarity, we interchangeably use the notation $p_{RC}$ instead of $p_{RC}(\omega)$ ($p_{RD}$, respectively, for $p_{RD}(\omega)$).
8.1 System Parameters and Evaluation Metrics

Degree of Autonomy

As intended by SOAs, services are supposed to be loosely coupled, thus executed autonomously. Enlisting participants to an atomic transaction, demanding their execution to rely on a coordinator’s decisions, limits the autonomy. The greater the autonomy, the more elements are actually loosely coupled. We define the degree of autonomy of a workflow $\omega$ as follows:

**Definition 22. Degree of Autonomy $d(\omega)$ of a Workflow $\omega$**

The degree of autonomy $d(\omega)$ of a workflow $\omega$ denotes the ratio of autonomous elements (i.e., non-coordinated elements) in $\omega$. Let $n$ denote the total number of elements in $\omega$, let $m = |M|$ be the number of coordinated elements. The degree of autonomy of $\omega$ is defined as:

$$d(\omega) := \frac{n - m}{n}$$

*Example:* For example, if $\omega$ consists of 10 elements, two of which are coordinated in a subtransaction, the degree of autonomy is $d(\omega) = \frac{10 - 2}{10} = 0.8$.

As the degree of autonomy depends on the approach chosen to execute a workflow we use a subscript to distinguish between the different approaches in the following.

The degree of autonomy evaluates an adapted workflow. In the according experiments, workflows are randomly generated according to the input parameters and adapted using AWM. By analyzing the altered workflow, the degree of autonomy is determined. However, as the autonomy is independent of the workflow’s execution we disregard the execution in the according experiments.

**Analytical Approach** In the following, we a concise analytical approach to estimate the degree of autonomy for AWM and WS-AT denoted as $d_{AWM}$ and $d_{AT}$ respectively. We apply this to verify our simulation results.

WS-AT does not explore properties of services. A workflow $\omega$ which is coordinated using WS-AT therefore requires all of its participants to be executed according to 2PC. Thus, all underlying resources of all participants have to be blocked until the coordinator propagates its decision. The resulting degree of autonomy applying WS-AT is hence:

$$d_{AT}(\omega) = 0$$ (8.1)

For our approach AWM, the degree of autonomy $d_{AWM}$ is determinable employing Theorem 1 (cf. Chapter 6): If there exist at least two conflicting elements, the set $M$ which needs to be coordinated is $M := E_{CP} \cup E_{Z}$. In the following, let $s(\omega_n)$ denote the expected size of $M$, where $\omega$ is a workflow of size $n$. Let further $r(\omega_n)$ be the expected
8.1 System Parameters and Evaluation Metrics

relative size of $M$, and $d(\omega_n)$ the thereby expected degree of autonomy. Due to the
definition of $d_{AW}$ the following then holds: $d(\omega_n) = 1 - r(\omega_n) = 1 - \frac{1}{n} \ast s(\omega_n)$.

Considering workflows without data dependencies, no directed transactional conflicts exist. Thus, $E_{CP} = \emptyset$ and $M = E_Z$, if $|E_Z| > 1$. The expected size $s(\omega_n)$ of $M$ in the case $n > 1$ complies with the number of non-recoverable, non-reordable elements:

$$s(\omega_n) = (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) \ast n, \text{ for } n > 1 \quad (8.2)$$

Therefore, the expected degree of autonomy is:

$$d(\omega_n) = 1 - r(\omega_n) = 1 - (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) \quad (8.3)$$

Example: Consider for example a workflow $\omega = WP_{AND}(s_1, \ldots, s_{12})$ consisting of 12 parallel aligned services. Assume that three $s \in \{s_1, \ldots, s_{12}\}$ are recoverable (i.e., $p_T(s) = (1, 0)$), three are redordable (i.e., $p_T(s) = (0, 1)$), three of them are both ($p_T(s) = (1, 1)$), and the remaining three expose $p_T(s) = (0, 0)$. Thus, $p_{RC}(\omega) = p_{RD}(\omega) = 0.5$. According to equation 8.3, the expected degree of autonomy for $\omega$ is then:

$$d(\omega_{12}) = 1 - r(\omega_{12}) = 1 - ((1 - 0.5) \ast (1 - 0.5)) = 0.75$$

Since exactly three elements expose $p_T(s) = (0, 0)$, these three need to be coordinated in a subtransaction $WP_{subTA}$. The other elements (75%) are executed before the $WP_{subTA}$ pattern if they are recoverable; the redordable services are executed afterwards.

Considering workflows with data dependencies, the number of coordinated elements grows by the directed transactional conflict elements $e \in E_{CP}$. The chance that an element $e$ is part of a directed transactional conflict is dependent on its transactional properties and the existing data dependencies.

We recursively determine the size of $M$, $s(\omega_n)$, for workflows which consist of a sequence of $n$ ($n \geq 1$) elements which are successively data dependent on their predecessor (i.e., $e_i \rightarrow e_{i+1} \rightarrow e_{i+2} \ldots$). For the sake of simplicity, we do not explicitly consider indirect conflict elements in the analytical approach. In the following, $M' = M \cup E_I$ denotes the set of coordinated elements $M$, unified with the set of indirect conflict elements $E_I$.

For the recursion base, we consider workflows of the form $\omega = WP_{SEQ}(s_1, s_2)$ in which $s_2$ is data dependent on $s_1$ (i.e., $s_1 \rightarrow s_2$). $s_1$ and $s_2$ directly transactionally conflict in case $s_1$ is not recoverable and $s_2$ is not redoable, i.e., the chance is: $(1 - p_{RC}(\omega)) \ast (1 - p_{RD}(\omega))$. The expected size of $M'$ is then:

$$s'(\omega_2) = 2 \ast (1 - p_{RC}(\omega_2)) \ast (1 - p_{RD}(\omega_2))$$

Accordingly, $s'(\omega_n)$ denotes the size of $M'$, $r'(\omega_n)$ the relative size of $M'$ and $d'(\omega_n)$ the according degree of autonomy. As $M \subseteq M'$, $d'(\omega_n)$ is a lower bound for $d(\omega_n)$, i.e., $d(\omega_n) \geq d'(\omega_n)$.  

---

4 Accordingly, $s'(\omega_n)$ denotes the size of $M'$, $r'(\omega_n)$ the relative size of $M'$ and $d'(\omega_n)$ the according degree of autonomy. As $M \subseteq M'$, $d'(\omega_n)$ is a lower bound for $d(\omega_n)$, i.e., $d(\omega_n) \geq d'(\omega_n)$. 

---

94
Appending a third element $s_3$ to $\omega$, if $M'$ is non-empty, it either consists of the same conflicting elements as determined for $\omega_2$ ($M' = \{s_1, s_2\}, |M'| = 2$), thus $s_3$ does not transactionally conflict. Or else, the newly appended element $s_3$ conflicts with $s_1$ (thus $M' = \{s_1, s_2, s_3\}, |M'| = 3$) or with $s_2$, however not with $s_1$ (thus $M = \{s_2, s_3\}, |M'| = 2$). Therefore $s'(\omega_3)$ is determined as:

$$s'(\omega_3) = s'(\omega_2) \ast p_{RD}(\omega_3) + 3 \ast (1 - p_{RC}(\omega_3)) \ast (1 - p_{RD}(\omega_3)) + 2 \ast p_{RC}(\omega_3) \ast (1 - p_{RC}(\omega_3)) \ast (1 - p_{RD}(\omega_3))$$

Generalizing this to a sequence of $n - 1$ services and appending the $n$th service, $M'$ either consists of conflicts which may occur in the subsequence $s_1$ to $s_{n-1}$ (without $s_n$ conflicting with any element) or all conflicts that may occur with the $n$th element. Therefore, $s'(\omega_n)$ is recursively determined as:

$$s'(\omega_n) = s'(\omega_{n-1}) \ast p_{RD}(\omega_n) + n \ast (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) + (n - 1) \ast p_{RC}(\omega_n) \ast (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) + (n - 2) \ast p_{RC}(\omega_n)^2 \ast (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) + \ldots + 2 \ast p_{RC}(\omega_n)^{n-2} \ast (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n))$$

This is simplified as follows:

$$s'(\omega_n) = s'(\omega_{n-1}) \ast p_{RD}(\omega_n) + \sum_{i=2}^{n} i \ast p_{RC}(\omega_n)^{n-i} \ast (1 - p_{RC}(\omega_n)) \ast (1 - p_{RD}(\omega_n)) \quad (8.4)$$

The relative size of $M'$ and the according degree of autonomy is again determined by:

$$d'(\omega_n) = 1 - r'(\omega_n) = 1 - \frac{1}{n} \ast s'(\omega_n) \quad (8.5)$$

In our experiments, we use Equations 8.3 and 8.4 to approximate the size of $M'$ (and thereby the degree of autonomy) for workflows which consist of a combination of elements: Those, which are data dependent on others, and those which are not.

**Semi-Atomicity Probability**

As previously stated, the autonomy of services is increased by omitting the coordination of elements. However, this may jeopardize the correctness of the execution of $\omega$. The
semi-atomicity probability of a workflow $\omega$ denotes the chance that execution of $\omega$ results in a correct, i.e. semi-atomic, termination. The greater the semi-atomicity probability, the more likely the workflow results in correct termination.

**Definition 23. Semi-Atomicity Probability $p_{SA}$ of a Workflow $\omega$**

The semi-atomicity probability $p_{SA}$ of a workflow $\omega$ is defined as the probability that $\omega$ semi-atomically terminates.

**Example:** Consider $\omega$ to be the sequence of two services $s_1$ and $s_2$, i.e., $\omega = WP_{SEQ}(s_1, s_2)$. Let $s_1$ and $s_2$ both be not recoverable and not retrievable: $p_T(s_1) = p_T(s_2) = (0, 0)$. Assume the probability for $s_1$ and for $s_2$ to successfully complete to be $p_S(s_1) = p_S(s_2) = 0.5$. $\omega$ semi-atomically terminates in case of failure of $s_1$ or both services complete successfully, thus: $p_{SA}(\omega) = (1 - 0.5) + 0.5 * 0.5 = 0.75$. Hence, in 75% of all cases, $\omega$ terminates correctly.

Note that $p_{SA}$ refers to the execution of a workflow. In the according experiments, generated workflows are executed in order to experimentally determine $p_{SA}$ employing WS-BA (denoted as $p_{SA}(\omega)_{BA}$). After that, the workflow is adapted and executed employing AWM to experimentally determine $p_{SA}$ (denoted as $p_{SA}(\omega)_{AWM}$).

**Analytical Approach** We provide an analytical approach to determine $p_{SA}$ in order to verify our simulation results. Note that AWM guarantees semi-atomic termination, as proven in Section 6.3.4.2, thus $p_{SA}(\omega)_{AWM}$ is 1 for all workflows $\omega$:

$$p_{SA}(\omega)_{AWM} = 1$$ (8.6)

For the sake of completion, we provide an analytical approach to determine $p_{SA}(\omega)_{BA}$ (for WS-BA). To simplify the analysis, we assume the success probability to be equal for all services in a workflow, i.e., $\forall s_i, s_j: p_S(s_i) = p_S(s_j) = p_S$. Let us first consider workflows which solely consist of $n$ parallel arranged services, $\omega = WP_{AND}(s_1, \ldots, s_n)$. Let $c$ services (i.e., $s \in \{s_{j1}, \ldots, s_{jc}\}$, with $0 \leq c \leq n$) be non-recoverable and all other services be recoverable, i.e., $c = (1 - p_{RC}(\omega)) * n$. $\omega$ terminates semi-atomically, if either all services complete or, in case of failure, all $c$ non-recoverable services. We refer to these failures as semi-atomic failures, since they allow for backward-recovery to preserve semi-atomicity. Let $p_S$ denote the success probability of all involved services (i.e., $p_S = p_S(s_i)$). Therefore:

---

\(^5\)As opposed to $d(\omega)$, which refers to an adapted workflow disregarding its execution.

\(^6\)This is sufficient, due to the assumption of homogeneous success probabilities in our experiments.
8.2 Empirical Evaluation of AWM

\[ p_{SA}(WP_{AND}(s_1, \ldots, s_n), c)_{BA} = \begin{cases} 1 & \text{if } c = 0 \\ p^n_s + (1 - p_s)^c & \text{if } c \geq 1 \end{cases} \]  \hspace{1cm} (8.7)

**Example:** Let, the number of elements be \( n = 10 \) and \( c = 2 \) of these services be non-recoverable. Let further the success probability of involved services be \( p_S = p_S(s_i) = 0.8 \). The semi-atomicity probability \( p_{SA} \) of this workflow is:

\[ p_{SA}(WP_{AND}(s_1, \ldots, s_{10}), 2)_{BA} = 0.8^{10} \cdot (1 - 0.8)^2 \approx 0.15. \]

Thus the chance, that \( \omega \) semi-atomically terminates is 15%. \( \blacklozenge \)

Regarding workflows with **sequentially aligned services**, i.e., workflows of the form \( \omega = WP_{SEQ}(s_1, \ldots, s_n) \), the semi-atomicity probability \( p_{SA} \) is determined as follows: Let again \( c \) services be non-recoverable (i.e., \( s \in \{s_{j1}, \ldots s_{jc}\} \), \( 0 \leq c \leq n \)) and \( s_{c0} \) for \( c_0 = \min\{j_1, \ldots j_c\} \) be the first non-compensatable service in the sequence (if \( c > 0 \)). Then, \( \omega \) semi-atomically terminates, if either all services complete or semi-atomic failure occurs: That is, in case of failure of \( s_{c0} \) or any other service aligned prior \( s_{c0} \). In these cases, \( \omega \) can be backward-recovered by recovering all completed services. Thus

\[ p_{SA}(WP_{SEQ}(s_1, \ldots, s_n), \{j_1, \ldots j_c\})_{BA} = p^n_s + (1 - p_s) \]

\[ + p_s \cdot (1 - p_s) + p_s^2 \cdot (1 - p_s) + \ldots + p_s^{c_0-1} \cdot (1 - p_s) \]

This is simplified as:

\[ p_{SA}(WP_{SEQ}(s_1, \ldots, s_n), \{j_1, \ldots j_c\})_{BA} = p^n_s + \sum_{i=0}^{c_0-1} p_s^i \cdot (1 - p_s) \]  \hspace{1cm} (8.8)

For sequential alignments, the absolute number of non-recoverable elements \( c \) is not decisive rather than the index \( c_0 \) of the first non-recoverable element. For an analytical approach to approximate \( c_0 \) for given \( c \) and \( n \), see Appendix D.1.

**8.2 Empirical Evaluation of AWM**

The aim of this section is to present the results of our experimental evaluation of AWM in a variety of system settings. Therefore, we vary all influencing parameters and illustrate the results classified according to the discussed metric (\( d(\omega) \) and \( p_{SA}(\omega) \)).
8.2 Empirical Evaluation of AWM

8.2.1 Autonomy of Participants $d_{AWM}$

In this section, we present the guaranteed degree of autonomy $d_{AWM}$ of AWM and evaluate the influence of different parameter settings on $d(\omega)$. Concurrently, we compare the results obtained for AWM with those for WS-AT. As WS-BA may produce incorrect system states, we do not consider its degree of autonomy.

A workflow $\omega$ which is coordinated using WS-AT requires all of its participants to be executed according to 2PC. Thus, all underlying resources of all participants have to be blocked until the coordinated propagates its decision. The resulting degree of autonomy applying WS-AT is $d_{AT}(\omega) = 0$ for all test cases $\omega$.

Our claim is that the degree of autonomy of AWM, $d_{AWM}(\omega)$ is higher than that of WS-AT in almost all cases. Only in the worst case, it is as low as that of WS-AT:

$$d_{AWM}(\omega) \geq d_{AT}(\omega)$$

In each test case, we generate at least 100 workflows and transform them using AWM. We quantify $d_{AWM}(\omega)$ by inspecting the resulting workflow, i.e., the elements within the WP$_{subTA}$ pattern with and without indirect conflict elements. The latter one is denoted as $d'_{AWM}$. We use the analytical approach to verify the experimental results.

The results are grouped according to various series of tests in which the influential parameters number of included elements, ratios of transactional properties, and present data dependencies are varied.

8.2.1.1 Number of Included Services $n$

Our analytical approach as well as the performed experiments show, that $d_{AWM}(\omega)$, as well as $d_{AT}(\omega)$, do not depend on the size of the workflow $n$: $d_{AWM}(\omega)$ is by a constant greater than $d_{AT}(\omega)$. We therefore forego the presentation of this series of tests and further omit the presentation of results for $d_{AWM}(\omega)$ varying $n$ in the following.

8.2.1.2 Ratios of Transactional Properties of Included Services

In this series of tests, we investigate the influence of the ratios of services which are recoverable $p_{RC}(\omega)$ and redoable $p_{RD}(\omega)$ on the degree of autonomy. In the performed experiments, the ratios are either both normally distributed around the same mean (i.e., $p_{RC}(\omega) = p_{RD}(\omega)$) or one of them remains fixed, while the other one is varied (e.g., $p_{RC}(\omega) = 0.5$ and $p_{RD}(\omega) = (0.0, 0.1, \ldots, 1.0)$).

**Test Set-Up** We perform the experiments on a workflow without data dependencies, and on two workflows with different mannered data dependencies. As $n$ does not influence $d_{AWM}$ (or $d_{AT}$), we choose $n = 12$ for all test workflows. We present results for the following types of workflows:

---

7The interested reader is referred to Appendix D.2.
• $\omega'$ is a workflow without data dependencies, consisting of 12 services, thus:

$$\omega' = WP_{\text{AND}}(s_1, \ldots s_{12})$$

• $\omega''$ consists of 12 services; for each set-up, we randomly choose six of them to be sequentially data dependent on the predecessor. The other six elements are randomly aligned in parallel or sequence. Thus, if the following data dependencies are randomly created $s_1 \rightarrow \ldots \rightarrow s_6$, an alignment of $\omega''$ is:

$$\omega'' = WP_{\text{AND}}(WP_{\text{SEQ}}(s_1, \ldots, s_6), s_7, \ldots, s_{12}).$$

• $\omega'''$ consists of 12 services; 4 of these are aligned in two sequences (e.g., $WP_{\text{SEQ}}(s_1, s_2)$ and $WP_{\text{SEQ}}(s_3, s_4)$) due to data dependencies (i.e., in this case $s_1 \rightarrow s_2$ and $s_3 \rightarrow s_4$). All other elements are randomly aligned in parallel or sequence. Thus, one characteristic alignment of $\omega'''$ is:

$$\omega''' = WP_{\text{AND}}(WP_{\text{SEQ}}(s_1, s_2), WP_{\text{SEQ}}(s_3, s_4), s_5, \ldots, s_{12}).$$

The results of the experiments with fixed values for $p_{\text{RD}}(\omega)$ and varied $p_{\text{RC}}(\omega)$ expose identical results (vice versa); therefore we omit their presentation.

**Assumption** According to our analytical model, we assume that $p_{\text{RC}}(\omega)$ and $p_{\text{RD}}(\omega)$ do not influence $d_{\text{AT}}(\omega)$. However, they strongly influence $d_{\text{AWM}}(\omega)$ for all three types of workflows: Obviously, the greater $p_{\text{RC}}(\omega)$, the greater the autonomy of included services. However, according to our analytical model, we assume $p_{\text{RD}}(\omega)$ to similarly influence $d_{\text{AWM}}(\omega)$. For a workflow without data dependencies, $\omega'$, $d_{\text{AWM}}(\omega')$ presumably converges as stated in Equation 8.3 to

$$d_{\text{AWM}}(\omega') = 1 - (1 - p_{\text{RC}}(\omega')) \times (1 - p_{\text{RD}}(\omega'))$$

Regarding workflows with inherent data dependencies $\omega''$ and $\omega'''$, the number of elements which are coordinated are increased by the directed transactional conflict elements. Thus, we assume $d_{\text{AWM}}(\omega')$ to be greater than $d_{\text{AWM}}(\omega'')$ and $d_{\text{AWM}}(\omega''').$

**Evaluation** In Figure 8.2 on the left side, $d_{\text{AWM}}(\omega')$ and $d_{\text{AT}}(\omega')$ are depicted on the y-axis. The results for experiments with fixed ratios of $p_{\text{RC}}$ (i.e., $p_{\text{RC}}$ equals 0.2, 0.5, and $p_{\text{RD}}$) and varied $p_{\text{RD}}$ values on the x-axis are illustrated.

It becomes apparent that $d_{\text{AWM}}(\omega')$ greatly depends on $p_{\text{RC}}$ and $p_{\text{RD}}$. If $p_{\text{RC}} = p_{\text{RD}} = 0$, the autonomy is $d_{\text{AWM}}(\omega') = 0$ as well, thus the same as for WS-AT. For greater values of $p_{\text{RC}}$ and $p_{\text{RD}}$, AWM performs significantly better than WS-AT. As soon as $p_{\text{RC}} = 1$ (or $p_{\text{RD}} = 1$), $d_{\text{AWM}}(\omega') = 1$, hence no coordination is needed. In these cases (in which no directed transactional conflicts occur), the size of $M$ tends to $(1 - p_{\text{RC}}) \times (1 - p_{\text{RD}}) \times n$. Our assumptions are thereby confirmed.

In Figure 8.2 on the right side, we depicted the results for $\omega''$ (solid lines) and $\omega'''$ (dashed line) for the experimental set-up $p = p_{\text{RC}} = p_{\text{RD}}$. As $\omega''$ contains a sequence of
8.2 Empirical Evaluation of AWM

![Graph](image)

Figure 8.2: $d_{AWM}$ for $\omega'$ (left) and $\omega''$, $\omega'''$ (right) varying $p_{RD}$.

...data dependencies of size greater than 2, indirect conflict elements occur. We illustrated the obtained degree of autonomy by AWM ($d_{AWM}(\omega)$) as well as the gained autonomy, when disregarding indirect conflict elements from the subtransaction (denoted as $d'_{AWM}(\omega)$). $d'_{AWM}(\omega''')$ is slightly lower than or equal to $d_{AWM}(\omega'')$ in this setting. Their deviation is in all cases less than 2%.

As can be seen, $d_{AWM}(\omega''') = d_{AWM}(\omega'')$ for $p = 0$ and $p = 1$. In all other cases, $d_{AWM}(\omega''') > d_{AWM}(\omega'')$. This is due to fewer data dependencies (i.e., $s_i \rightarrow s_j$) being present in $\omega'''$. Furthermore, the ratio of elements restricted by data dependencies is lower in $\omega'''$ ($r = 1/3$ in $\omega'''$ vs. $r = 1/2$ in $\omega''$). Therefore, the inherent conflict potential for directed transactional conflicts is lower for $\omega'''$ than for $\omega''$.

In Figure 8.3, the results for constant values of $p_{RD} = 0.2, 0.5$ are depicted for $\omega''$ (solid lines) and $\omega'''$ (dashed lines). If only half of the elements are recoverable (i.e., $p_{RC} = 0.5$), the resulting $d_{AWM}(\omega')$ increases from roughly 0.35 ($p_{RC} = 0$) to 1, thus decreasing the number of coordinated elements up to 65% as opposed to WS-AT. For $p_{RC} = 0.2$ the resulting $d_{AWM}(\omega'')$ increases from roughly 0.12 for $p_{RD} = 0$ to 1. Again, for all values of $p_{RC}$ and $p_{RD}$, $d'_{AWM}(\omega''')$ is greater than $d'_{AWM}(\omega'')$. Applying the analytical approach to approximate $d'_{AWM}$, our results are verified (see Appendix D.3).

![Graph](image)

Figure 8.3: $d_{AWM}$ of $\omega''$ and $\omega'''$ with data dependencies, varying $p_{RD}$.
8.2 Empirical Evaluation of AWM

Conclusion As assumed, the ratios $p_{RC}$ and $p_{RD}$ significantly influence the resulting size of the subtransaction and thus the degree of autonomy $d_{AWM}$. Obviously, the more elements are recoverable, the greater $d_{AWM}$ (if all elements are recoverable, no coordination is needed just as with WS-BA). Furthermore, $p_{RD}$ similarly influences $d_{AWM}$: The more elements are redoable, the more elements guarantee that they eventually complete, thus fewer elements have to be coordinated. Generally, if at least one element which is recoverable or redoable exists, the resulting $d_{AWM}$ is greater than $d_{AT}$. In the presented experiments, AWM increases the autonomy of elements up to 100% (e.g., if $p_{RC} = 1$ or $p_{RD} = 1$) as opposed to WS-AT.

Decisive for the size of $M$ is the product of the complements of $p_{RD}$ and $p_{RD}$: $(1 - p_{RC}) \times (1 - p_{RD})$. It denotes the chance that two elements which are data dependent on each other, transactionally conflict (i.e., the chance for $e_i, e_j \in E_{CP}$), as well as the probability, that an element is neither retrieable nor redoable in a given setting, thus $e \in E_Z$. In this series of test, the simulation results match the expected values derived from the analytical approximation.

8.2.1.3 Data Dependencies

Test Set-Up In this series of tests, we investigate the influence of data dependencies on the resulting degree of autonomy of a workflow. First of all, we vary the ratio $r$ of elements which are data dependent on others. We additionally vary the average length $l$ of data dependencies (cf., $s_{i1} \rightarrow s_{i2} \rightarrow \ldots \rightarrow s_{il}$). Furthermore, we vary the average number $t$ of these data dependency sequences.

We present results of experiments with workflows of size $n = 20$ and homogeneous ratios of transactional properties ($p_{RC} = p_{RD}$), normally distributed around $p = 0.5$.

Assumption With an ascending ratio $r$, the number of elements involved in data dependencies and thus the prevailing conflict potential increases. We therefore assume the degree of autonomy $d_{AWM}$ to decrease with increasing $r$.

Furthermore, when increasing number of dependency sequences $t$ we assume $d_{AWM}$ to ascend, for fixed values of $r$. We base this on the following consideration:

The less data dependencies (of greater length $l$, since $r$ remains fixed) exist, the more dependencies of the form $s_i \rightarrow s_j$ exist in the workflow. In the extreme case (for $r = 1$), if $t = n/2$ sequences occur, each of them of length $l = 2$, $n/2$ data dependencies exist. On the other hand, if $\omega$ consists of only one data dependency sequence ($t = 1$, with $r = 1$), that is $s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_n$, more data dependencies, namely $n - 1$ are present in the workflow. Thus, in the latter case ($t = 1$), the conflict potential inherent to $\omega$ is lower than for more sequences $t$ (of shorter length $l$).
8.2 Empirical Evaluation of AWM

Evaluation In Figure 8.4, \(d_{AWM}(\omega), d'_{AWM}(\omega),\) and \(d_{AT}\) are depicted, varying the length of the data dependency sequences \(l\). We illustrate the results for series of tests with one, two and four data dependency sequences \((t = 1, 2, 4)\). For each sequence, we vary its length \(l\) from 2 to the maximal possible value \((l = 1, \ldots, n/s)\). The remaining \(n - l \cdot t\) elements are appended randomly in sequence or parallel.

Figure 8.4: \(d_{AWM}(\omega)\) with \(t = 1, 2, 4\) sequences varying length \(l\).

For all three series of tests it can be seen, the greater \(l\) (thus the longer the data dependency sequences) the lower the resulting \(d_{AWM}(\omega)\) (and \(d'_{AWM}(\omega)\)). This is due to the fact, that the greater \(l\), the greater the ratio of elements \(r\) which are involved in data dependencies. If \(l = 0\), all obtained degrees of autonomy are 0.75 (\(d_{AWM}(\omega) = d'_{AWM}(\omega) = 0.75\)), which corresponds to the proportion of non recoverable and non redoable elements \((1 - p_{RC}) \cdot (1 - p_{RD})\). For \(l > 2\), \(d_{AWM}(\omega) > d'_{AWM}(\omega)\) for all \(t\). With ascending \(l\), more indirect conflict elements exist, which are excluded from coordination using AWM (as opposed to AWM'). Thus, the greater is the benefit from exploitation of indirect conflict elements (up to 20% for \(t = 1\) and \(l = 20\)).

Again, for all series of test, the resulting \(d_{AT}(\omega)\) is zero, thus lower than \(d_{AWM}(\omega)\). In regard to the autonomy, one benefits from employing AWM as opposed to WS-AT by \(\sim 30\%\) (for \(t = 1\) and \(l = 10\)) to approximately 75\% \((l = 2)\).

Comparing these tests, it can be seen, that the more data dependency sequences exist within the workflow (thus, the greater \(s\)), the lower the resulting \(d_{AWM}(\omega)\) (and \(d'_{AWM}(\omega)\)). However, this misleading insight is a result the set-up:

If only one sequence of length 4 (thus \(t = 1\) and \(l = 4\)) exists, the ratio of data dependent elements in \(\omega\) is \(r = 1/5\). On the other hand, if four sequences of length four exist (thus \(t = l = 4\)), the resulting ratio is higher \(r = 4/5\). Therefore, no direct comparison can be drawn between the series of test. In order to further investigate the influence of \(t\) and \(l\), we examine the results depicted in Figure 8.5.

This time, we varied the number \(t\) of sequences (on the x-axis) while keeping the ratio of data dependent elements \(r\) constant. In Figure 8.5, the resulting \(d_{AWM}(\omega)\) and \(d'_{AWM}(\omega)\) are shown for \(r = 0.4, 0.7\) and 1.0. When varying the number of sequences
for a fixed ratio $r$, it holds that at most $t = r \times n/2$ data dependency sequences are present, as they always consist of at least two elements.

![Graph](image)

**Figure 8.5:** $d_{AWM}(\omega)$ varying the number of sequences $t$, with fixed ratios $r$.

Obviously, the greater the ratio of data dependent elements $r$, the lower the resulting $d_{AWM}(\omega)$. This is due to the ascending conflict potential with increasing $r$. Additionally, it can be seen that all values of $d_{AWM}(\omega)$ (and $d'_{AWM}(\omega)$) increase with ascending $t$ to the maximum value 0.75. In this case, the set of elements which needs to be coordinated consists of all non-recoverable and non-redoable elements $((1 - p_{RC}) \times (1 - p_{RD}))$ and all conflicting pairs $((1 - p_{RC}) \times (1 - p_{RD}))$. Due to the chosen ratios of recoverable and redoable elements $p_{RC} = p_{RD} = 0.5$, $d_{AWM}(\omega)$ (and $d'_{AWM}(\omega)$) converge to 0.75.\(^8\)

The interesting perception of this test is that if the $r \times n$ data dependent elements are arranged in few sequences $t$ (with greater length $l$), the resulting $d_{AWM}(\omega)$ (and $d'_{AWM}(\omega)$) is lower than for more (but shorter) sequences. This affirms our assumption: For longer sequences (greater $l$) the number of data dependencies of the form $s_i \rightarrow s_j$ increases, thus more conflicts occur, i.e., the autonomy decreases.

Finally, we want to point out that with greater length of data dependency sequences $l$ (in this case synonymous with lower values $t$), the difference between $d_{AWM}(\omega)$ and $d'_{AWM}(\omega)$ increases: The benefit by disregarding indirect conflict elements from coordination ascends. For $r = 1.0$, the difference accounts roughly 20% ($t = 1$), for $r = 0.4$, $d_{AWM}$ is approximately 5% greater than $d'_{AWM}$.

**Conclusion**  Obviously, by increasing the ratio of data dependent elements $r$, the conflict potential is increased, thus resulting in lower degrees of autonomy $d_{AWM}(\omega)$ (as assumed). On the other hand, we are able to verify, that for fixed values of $r$, the degree of autonomy $d_{AWM}(\omega)$ is increased, if more sequences $t$ of shorter length $l$ are inherent to the workflow. Thereby, the absolute number of data dependencies decreases, thus lowering the $\omega$ inherent conflict potential. Further, we want to emphasize, that especially in the case of long data dependency sequences, the benefit from excluding indirect

\(^8\)We verified this, using different ratios $p_{RC}$ and $p_{RD}$.  

103
conflict elements from coordination is substantial. In this series of tests, the benefit from employing AWM as opposed to WS-AT ranges from ca. 10% \((t = 1\text{ and } r = 1)\) up to 75% (for all values of \(r\) and \(t = l \ast n/2\)).

Overall, in Subsection 8.2.1.1 through 8.2.1.3, we are able to verify our claim, that the resulting degree of autonomy \(d_{AWM}(\omega)\) (and \(d'_{AWM}(\omega)\)) are greater than or equal to \(d_{AT}(\omega)\) in all proposed settings. Thus, regarding the autonomy granted to participants, AWM outperforms the pessimistic approach WS-AT.

### 8.2.2 Correctness Guarantees \(p_{SA}\)

In this section, we evaluate the resulting correctness guarantees of AWM and compare them to the results of an optimistic approach to support transactional execution, i.e., WS-BA. We investigate the influence of different parameter settings on \(p_{SA}\). As the pessimistic approach WS-AT guarantees correctness in all cases, just as AWM, we omit the presentation of results for WS-AT.

In order to draw a fair distinction between AWM and WS-BA, we employ the notion of semi-atomicity (see Section 5.3) to define correct execution for WS-BA. That is, the execution of a workflow \(\omega\) is correct employing WS-BA, if it either successfully completes, or all elements which demand consistent closure (i.e., all \(e\) with \(p_{CT}(e) = (\ast, 1, \ast)\)) are compensated in case of failure.\(^9\)

Our claim for this section is, that the semi-atomicity probability using AWM is greater than or equal to that using WS-BA in any case. Especially, as AWM ensures correctness in any case, we assume \(p_{SA}(\omega)_{AWM}\) to be 1 for all series of tests:

\[
1 = p_{SA}(\omega)_{AWM} \geq p_{SA}(\omega)_{BA}
\]

Workflows are generated according to the input parameters and executed at least 100 times to determine the ratio of executions that terminated semi-atomically using AWM and WS-BA. We use the analytical approach to verify our experimental results.

The results are grouped according to the series of tests in which the influential parameters number of included elements \(n\), ratio of recoverable elements \(p_{RC}(\omega)\) and success probability of services \(p_S\) are varied. Regarding the success probability, we assume homogeneous environments. Thus, in our experiments, we randomly choose the success probability of all included services normally distributed around a stated mean \(p_S\) with a standard deviation of 5%.

As the initial alignment of services in different patterns influences \(p_{SA}\), we present results for two different types of workflows: On the one hand, we consider workflows in which services are aligned in parallel, i.e., \(\omega = WP_{AND}(s_1, \ldots, s_n)\). We additionally regard workflows, in which services are sequentially aligned: \(\omega = WP_{SEQ}(s_1, \ldots, s_n)\).\(^10\)

\(^9\)Thus, we regard the recoverability of services as opposed to compensatability.

\(^10\)Evaluation of realistic examples incorporating combinations of these are presented in Section 8.3.
8.2 Empirical Evaluation of AWM

8.2.2.1 Number of Included Services $n$

Considering our analytical approach to determine $p_{SA}(\omega)_{BA}$, we assume that $p_{SA}(\omega)_{BA}$ is not mainly determined by $n$: For parallel alignments, Formula 8.7 on page 97 conveys that $p_{SA}(\omega)_{BA}$ decreases with increasing $n$. However, Formula 8.8 hints, that in sequential alignments, $n$ is not substantially decisive for $p_{SA}(\omega)_{BA}$: It is mainly determined by the ratio of recoverable elements $p_{RC}(\omega)$ and success probability $p_S$. We therefore forego the presentation of this series of tests\textsuperscript{11} and further abandon from presenting results for $p_{SA}(\omega)_{BA}$ varying $n$.

8.2.2.2 Ratio of Recoverable Elements $p_{RC}$

**Test Set-Up** In this series of test, we vary the ratio of recoverable elements within a workflow from $p_{RC}(\omega) = 0$ to 1. We present results for parallel $WP_{AND}$ and sequential $WP_{SEQ}$ alignments of services for different values of $p_S$ and a fixed size of elements in the workflow of $n = 50$.

**Assumption** Employing the analytical approach to determine $p_{SA}(\omega)_{BA}$, we assume $p_{SA}(\omega)_{BA}$ to increase with an increasing ratio of recoverable elements $p_{RC}(\omega)$. That is, the more elements are actually recoverable, the more failure cases are recoverable and thus semi-atomic failures. However, the actual increase depends on the alignment of services as well as their success probability $p_S$.

**Evaluation** On the left hand side of Figure 8.6, the results for the parallel alignment $WP_{AND}$ for $p_S = 0.1$ and $p_S = 0.5$ are depicted. For small ratios of recoverable elements $p_{RC}(\omega)$, $p_{SA}(\omega)_{BA}$ is roughly 0. That is, correct termination in terms of semi-atomicity is the exceptional case employing WS-BA (as opposed to AWM, as $p_{SA}(\omega)_{AWM} = 1$). With increasing values of $p_{RC}(\omega)$, starting from roughly $p_{RC} = 0.5$, $p_{SA}(\omega)_{BA}$ quickly increases for both depicted values of $p_S$. For $p_{RC} = 1$, the resulting semi-atomicity probabilities for both depicted series of tests approach $p_{SA}(\omega)_{BA} = 1$. Note that even if only 10% of services are not recoverable (and $p_S = 0.5$), correctness guarantees given by WS-BA are lower than 10%.

Note also that for depicted experiments, $p_{SA}(\omega)_{BA}$ is lower for a higher success probability. This is due to the fact that with lower success probabilities of elements $p_S$, the chance of failures absolutely seen increases, and as well the probability that recoverable failures occur. However, this does not generally hold for all values of $p_S$. The influence of success probabilities is further discussed in Section 8.2.2.3.

On the right hand side of Figure 8.6, the results for sequential alignment $WP_{SEQ}$ are depicted for the same parameter settings ($p_S = 0.1$, $p_S = 0.5$). Again, $p_{SA}(\omega)_{BA}$ is

\textsuperscript{11} The interested reader is referred to Appendix D.4.
8.2 Empirical Evaluation of AWM

considerably lower than $p_{SA}(\omega)_{AWM}$ for small values of $p_{RC}(\omega)$. However, as opposed to the parallel alignment on the left hand side, it is clearly greater than 0, owing to the nature of the execution of $WP_{SEQ}$ patterns: As services are executed successively, failure before the $c_0$th element is backward recoverable, thus preserving semi-atomicity. The index of the first non-recoverable element $c_0$ is decisive for $p_{SA}(\omega)_{BA}$.

With an ascending ratio of recoverable elements, $p_{SA}(\omega)_{BA}$ increases for all values of $p_S$. In the presented experiments, $p_{SA}(\omega)_{BA}$ increases from approximately $p_{SA}(\omega)_{BA} \approx 0.5$ for $p_S = 0.5$ and $p_{RC}(\omega) = 0$ to $p_{SA}(\omega)_{BA} = 1$. The resulting $p_{SA}(\omega)_{BA}$ for workflows with the lower success probability ranges from $p_{SA}(\omega)_{BA} \approx 0.9$ for $p_S = 0.1$ and $p_{RC}(\omega) = 0$ to $p_{SA}(\omega)_{BA} = 1$. Using the experimental results for $c_0$, these values are verified by the analytical approach: For $n = 50$ and $p_{RC}(\omega) = 0.8$, the expected value for $c_0 \approx 3$. Using Formula 8.8 on page 97, if $p_S = 0.5$ the analytically determined correctness guarantee is $p_{SA}(\omega)_{BA} \approx 0.93$.

With an ascending ratio $p_{RC}(\omega)$, the expectation value for $c_0$ increases as well. The greater $c_0$, the more failure cases can be backward recovered to semi-atomically terminate. In the extreme case, if all included elements are recoverable, the execution of $\omega$ employing WS-BA guarantees semi-atomicity. Therefore, $p_{SA}(\omega)_{BA}$ increases with an ascending ratio of recoverable elements $p_{RC}$.

**Conclusion** This series of tests confirms our assumption that $p_{SA}(\omega)_{BA}$ is dependent on $p_{RC}(\omega)$ in the following way: The fewer elements are recoverable, the lower the resulting correctness probability $p_{SA}(\omega)_{BA}$. In parallel alignments, the influence of lower $p_{RC}(\omega)$ is stronger than for sequential workflows $WP_{SEQ}$. In parallel alignments a fairly low ratio of non-recoverable elements suffices to tremendously raise the risk of inconsistent termination: In the depicted experiments, if only 20% of the included elements are non-recoverable (i.e., $p_{RC}(\omega) = 0.8$), WS-BA guarantees correct execution in less than 30%
8.2 Empirical Evaluation of AWM

of cases (for both $p_S = 0.1$ and $p_S = 0.5$).

All results are verified applying the analytical approach, cf. Formulas 8.7 and 8.8. In the presented scenarios, the risk for inconsistent system states is higher for greater success probabilities of the elements. This is not generally true, and thus further investigated in the next section.

8.2.2.3 Success Probability of Services $p_S$

**Test Set-Up** In this series of tests, we investigate the influence of the success probability of single services on the resulting semi-atomicity probability $p_{SA}(\omega)_{BA}$. We therefore choose workflows with $n = 50$ elements and different ratios of recoverable elements $p_{RC} = 0.5$ and $p_{RC} = 0.9$.

**Assumption** According to the analytical approach, we assume the success probabilities of the elements to crucially influence the resulting $p_{SA}(\omega)_{BA}$. Regarding both Formulas 8.7 and 8.8 more closely, one sees that both consist of two terms: The first one denotes the success probability of the whole workflow, that is $\omega$ successfully completes. The second one refers to semi-atomic failures, which can be backward-recovered. We depicted these terms for a parallel alignment $WP_{AND}$ with $n = 10$ and $c = 5$ in Figure 8.7.

![Behavior of success (blue) and semi-atomic failure (red) of $WP_{AND}$ varying $p_S$.](image)

Consider these terms separately: For any alignment, the chance of semi-atomic failures (red curve in Figure 8.7) is high for low success probabilities $p_S$ (cf., it equals 1 for $p_S = 0$) and decreases with ascending values of $p_S$. This is due to the factor $(1 - p_S)$ present in both formulas. The chance for success of the whole workflow is in both alignments $p_S^n$ (blue line in Figure 8.7) thus increasing from 0 for $p_S = 0$ to 1 for $p_S = 1$. The sum of both yields to $p_{SA}(\omega)_{BA}$. We therefore assume, $p_{SA}(\omega)_{BA}$ to decrease from 1 (for $p_S = 0$) with ascending success probabilities to a global minimum. In these cases, $p_{SA}(\omega)_{BA}$ is mainly determined by the chance for semi-atomic failure (while the probability for successful completion is nearly 0). After that, we assume $p_{SA}(\omega)_{BA}$ to increase again to 1 (for $p_S = 1$), as the chance for successful completion increases and the chance for semi-atomic failure approaches 0.
Evaluation Considering the results of the $W_{P\text{AND}}$ alignment in Figure 8.8 on the left hand side, the resulting semi-atomicity probability (depicted on the y-axis) is 1 for $p_S = 0$ for both illustrated test series ($p_{RC} = 0.5$ and $p_{RC} = 0.9$). In this case, all services fail, which is a semi-atomic system state. For increasing $p_S$, $p_{SA}(\omega)_{BA}$ decreases to 0: In the case $p_{RC} = 0.5$, it quickly approaches 0 (at roughly $p_S = 0.2$), for $p_{RC} = 0.5$ it converges to 0 for $p_S = 0.5$. This behavior verifies our assumption that for low success probabilities, the chance of semi-atomic failures (quantified by $(1 - p_S)^k$) decreases. For higher ratios of recoverable elements $p_{RC}$, $k$ is lower, thus the decrease is more gentle. For higher success probabilities the resulting semi-atomicity probabilities of both depicted cases quickly increase to 1. That is due to the chance of successful completion of the whole workflow (determined by $p_S^n$) thus independent of $p_{RC}$. For the depicted value of $n$, this value does not noticeably increase until $p_S = 0.9$.\(^\text{12}\) In conclusion, in the depicted scenarios, AWM outperforms WS-BA, as it guarantees correctness in all cases thus partially increasing $p_{SA}$ by 100%.

For the sequential set-up $W_{P\text{SEQ}}$ depicted on the right hand side of Figure 8.8, the resulting values of $p_{SA}(\omega)_{BA}$ exhibit a similar behavior as for $W_{P\text{AND}}$. However, applying the same parameters, $p_{SA}(\omega)_{BA}$ does not decline to 0 rather than to a global minimum greater than 0. In this scenario ($n = 50$) the global minimum of $p_{SA}(\omega)_{BA}$ is reached at $p_S \approx 0.9$. This again relies on the fact, that for $n = 50$, $p_S^n$ does not noticeably increase until $p_S \approx 0.9$. For shorter sequences, the minimum of $p_{SA}(\omega)_{BA}$ is reached for lower success probabilities $p_S$.

The chance, that occurring failures are backward recoverable thus preserving semi-atomicity is considerably higher in the sequential set-up $W_{P\text{SEQ}}$ as opposed to $W_{P\text{AND}}$. In the depicted series of tests, the experimental evaluation of the average index of the first non-recoverable element is $c_0 \approx 2.38$ for $p_{RC} = 0.5$ and $c_0 \approx 8.07$ for $p_{RC} = 0.5$. Using these to determine $p_{SA}(\omega)_{BA}$ analytically applying Formula 8.8, the achieved results are verified.

\(^{12}\)For workflows with fewer elements, $p_{SA}(\omega)_{BA}$ noticeably increases for lower values of $p_S$. 

---

\[\text{Figure 8.8: } p_{SA} \text{ of an } W_{P\text{AND}} \text{ and } W_{P\text{SEQ}}, \text{ varying the success probability of services } p_S.\]
8.3 Evaluation of AWM in Realistic Settings

**Conclusion** These series of tests verify our assumption that $p_S$ decisively influences $p_{SA(\omega)BA}$. In all investigated scenarios, for small success probabilities, $p_{SA(\omega)BA}$ decreases from 1 (for $p_S = 0$) to a scenario specific minimum with increasing $p_S$. This minimum is determinable using the analytical approach (Formulas 8.7 and 8.8); it is dependent on the probability that semi-atomic failure occurs. Further increasing the success probability $p_S$ results in an increased semi-atomicity probability $p_{SA(\omega)BA}$ which approaches 1 for $p_S = 1$. This relies on the probability that all elements and thus the whole workflow successfully complete.

Overall, in all of the previous series of tests, we were able to verify our claim, that the resulting semi-atomicity probability $p_{SA(\omega)AWM}$ employing AWM is greater than or equal to $p_{SA(\omega)BA}$ in all investigated settings. The evaluation revealed, as soon as only a few non-recoverable elements are included in the workflow, $p_{SA(\omega)BA}$ may tremendously decrease. Thus, AWM is indispensable to guarantee correct execution.

8.3 Evaluation of AWM in Realistic Settings

In this section, we present evaluation results for AWM in two realistic example scenarios. At first, we provide the results for the running example MoP. Additionally, we evaluate AWM for another realistic, however more sophisticated, scenario. We employ the metrics degree of autonomy and semi-atomicity probability to quantify obtained results.

8.3.1 Evaluation Example I: MoP

**Test Set-Up** In this series of test, we investigate our running example scenario of MoP, which is depicted in Figure 1.2 on page 4. In this scenario, we assume the customers request specification $CRS$ and the ticket delivery $Confirm$ services to run on the vendors local machine. Due to their semantics, we assume both to be recoverable and redoable (i.e., $p_T(CRS) = p_T(Confirm) = (1, 1)$).

In the following, we vary the ratios of recoverable $p_{RC(\omega)}$ and redoable $p_{RD(\omega)}$ elements bound at runtime and then adapt the workflow using AWM to determine the resulting degree of autonomy $d_{AWM(\omega)}$. We additionally vary the success probability of services $p_S$ and execute the workflow a minimum number of 100 times to determine the resulting semi-atomicity probability $p_{SA}$.

Please note that depicted ratios $p_{RC(\omega)}$ and $p_{RD(\omega)}$ refer to the dynamically bound services, thus specifically exclude the local services $CRS$ and $Confirm$. The same holds for the depicted success probabilities of services $p_S$; The depicted values of $p_S$ refer solely to the dynamically bound services, i.e., $p_S(CRS) = p_S(Confirm) = 1$ in any case.

**Assumption** We base our assumptions on the analytical approaches to determine the degree of autonomy and the semi-atomicity probability (see Section 8.1) as well as the
results and conclusions of Section 8.2.

Regarding the degree of autonomy of AWM we assume the following: If all of the bound services are neither recoverable nor redoable, all 5 dynamically bound services have to be included in the set of coordinated elements $M$. In this case, the resulting degree of autonomy is: $d_{AWM}(\omega) = 1 - 5/7 \approx 0.29$. According to Section 8.2.1.2, with increasing $p_{RC}(\omega)$ and $p_{RD}(\omega)$, we assume $d_{AWM}(\omega)$ to steadily increase to $d_{AWM}(\omega) = 1$ (recall the results of Figure 8.2 in particular).

Considering the correctness of the workflow executing it employing WS-BA, we assume the following according to the analytical approach: If again none of the dynamically bound elements is recoverable, the workflow semi-atomically terminates, in case none of these elements or the whole workflow is completed. In the latter case (successful termination of $\omega$), all elements of the WP$_{AND}$ pattern have to complete as well as the WP$_{XOR}$ pattern:\footnote{These are Philharmonics, Transportation, Reservation, PayCC and PayCh.} $p_S^3 * (p_S + (1 - p_S) * p_S)$. In case of failure of $\omega$, all activated dynamic services have to fail as none of them is recoverable: $(1 - p_S)^3$. In conclusion, if the success probability of the dynamically bound services is e.g., $p_S = 0.9$, the probability that $\omega$ semi-atomically terminates using WS-BA is the sum of the aforementioned cases:

$$p_{SA}(\omega)_{BA} = 0.9^3 * (0.9 + (1 - 0.9) * 0.9) + (1 - 0.9)^3 \approx 0.72$$

Just as in Section 8.2.2.2, we assume $p_{SA}(\omega)_{BA}$ to rise to 1 with increasing ratio $p_{RC}$ (recall Figure 8.6 in particular). Increasing $p_S$, we assume (similar to Figure 8.8) $p_{SA}(\omega)_{BA}$ to be 1 (for $p_S = 0$), a decrease to a global minimum and after that an increase again to 1 (for $p_S(s_i) = 1$).

### Evaluation

In Figure 8.9, we depict the resulting degree of autonomy (on the y-axis) using AWM and WS-AT. On the x-axis, the ratio $p_{RC}(\omega)$ is varied from $p_{RC} = 0$ to $1$.\footnote{As we assume $p_S(\text{CRS}) = p_S(\text{Confirm}) = 1$, we omit them in the consideration.} We present the results for $p_{RD} = 0.5, 0.8$ and $p_{RD} = p_{RC}$. As assumed, the progress of the curve is similar to Figure 8.2.\footnote{As the results for varying $p_{RD}(\omega)$ are similar, we omit their presentation.}

Considering the case $p_{RD} = p_{RC}$, as analyzed in the assumption, $d_{AWM}$ raises from $d_{AWM}(\omega) \approx 0.29$ (for $p_{RC} = 0$) steadily to $d_{AWM}(\omega) = 1$ (for $p_{RC} = 1$). In this example scenario, AWM enormously increases the autonomy of included services: If solely half of the dynamically bound elements are recoverable and half of them are redoable, AWM enhances the autonomy by over 80%.

In Figure 8.10, the probabilities of semi-atomic termination of $\omega$ employing AWM and WS-BA are depicted. On the left hand side, we varied the ratio of recoverable elements $p_{RC}$ and show results for $p_S = 0.5$ and 0.9. AWM guarantees correct execution in any\footnote{As opposed to the referenced Figure 8.2, $d_{AWM}(\omega) > 0$ in all depicted cases, as we varied the ratio of dynamically bound services. As mentioned, the properties of the local services CRS and Confirm are not varied. Thus, as opposed to Figure 8.2, the depicted ratio of all elements ranges from $2/7$ to $1$.}
Chapter 8.3 Evaluation of AWM in Realistic Settings

Figure 8.9: $d_{AWM}$ and $d_{AT}$ of the MoP example, varying $p_{RC}$.

setting: $p_{SA}(\omega)_{AWM} = 1$. Using WS-BA, inconsistent system states may occur: If the success probability is $p_S = 0.5$, $p_{SA}(\omega)_{BA}$ is lowered to $\sim 0.5$ for $p_{RC} \leq 0.4$. Considering the two depicted cases for WS-BA, the greater success probability ($p_S = 0.9$) results in the greater $p_{SA}(\omega)_{BA}$. However, this does not generally hold, as $p_{SA}(\omega)_{BA}$ is dependent on $p_S$ as shown on the right side.

Figure 8.10: $p_{SA}$ of the MoP example, varying $p_{RC}$ and the success probability $p_S$.

The right part of Figure 8.10 depicts a variation of $p_S$ and the resulting $p_{SA}$ for $p_{RC} = 0.5, 0.9$. The curves exhibit similar trends as the general evaluation results (see Figure 8.6 on the right side and Figure 8.8). $p_{SA}(\omega)_{BA}$ decreases from 1 with increasing $p_S$ to the global minimum of $p_{SA}(\omega) \approx 0.6$ for $p_S = 0.4$. Further increasing $p_S$, $p_{SA}(\omega)_{BA}$ is raised to 1 (for $p_S = 1$). Our assumptions are thereby confirmed.

Conclusion The evaluation of the MoP scenario confirms our assumptions which we based on the analytical model as well as the evaluation results discussed in Section 8.2. It thereby nicely reflects the results obtained from the generic evaluation and hence validates its utility. Furthermore, this simple scenario exposes the considerable enhancement of AWM as opposed to the existing approaches WS-AT and WS-BA. In the displayed scenario, AWM outperforms WS-AT up to 70% regarding $d_{AWM}$. It ensures correct execution in any case, thus increasing $p_{SA}(\omega)_{BA}$ by up to 40% as well.
8.3.2 Evaluation Example II: Order-to-Delivery Process

In this section we evaluate a more sophisticated realistic scenario, namely the order-to-delivery process. As depicted in Figure 8.11, the provider operates an online marketplace, which works as follows.

![Diagram of the order-to-delivery process](image)

Figure 8.11: The order-to-delivery process.

After a customer specifies the desired products, offers are searched for at present providers. Next, if the customer is logged-in, she may confirm certain providers of the list. In parallel, the provider files the search results for analytical processing reasons. According to a customer’s specification, she is notified either via email or text message that the process’ state is confirmed. The marketplace contacts the vendors (Vendor...
V_1, \ldots, V_i\) and purchases the specified products. If the purchase successfully completes, the according information is again filed. In the next step, several delivery services are approached which deliver the product to the client \((Delivery S_1, \ldots, Delivery S_j)\). In parallel trusted payment providers (such as PayPal\(^{17}\)) are conducted to perform the payment \((Pay P_1, \ldots, Pay P_k)\). The customer is again notified if the current status of her purchase changes. The chosen delivery and payment providers are then rated. If all services complete successfully, the process information is archived.

In this scenario, all services within the \(WP_{\text{XOR}}\) patterns \((\text{XorVendor}, \text{XorDelivery} \) and \(\text{XorPay}\)) are dynamically bound at runtime. Their inclusion is dependent on the product search as well as on the offers of single vendors. We assume all other services to be locally deployed: The properties of local services remain fixed and are given as depicted in Figure 8.11. Data dependencies are displayed as gray dashed lines.

We group the presented results as follows: In the \textit{general evaluation}, autonomy and correctness guarantees are investigated while varying the established parameters \(p_{RC}(\omega)\), \(p_{RD}(\omega)\) and \(p_S(s_i)\). Afterwards, we presents results for the \textit{combination of AWM and service discovery} (recall Chapter 6.4): Here, we evaluate the effects of multiple service providers on the autonomy of services and the correctness guarantees for the workflow.

\subsection*{8.3.2.1 General Evaluation}

Evaluating the order-to-delivery process confirms the results achieved by the general evaluation (Section 8.2) as well as those of the MoP process (Section 8.3.1). We thus forego their presentation and refer to Appendix D.5. The results reveal, if only half of the included elements are redoable, AWM is able to increase the autonomy \(d_{AWM}(\omega)\) by approximately 90\% as opposed to WS-AT. Considering the correctness guarantee, WS-BA ensures correct execution in only 65\% of all cases, if the success probability of the bound elements is \(p_S = 0.5\) and only half of the dynamically bound elements are recoverable. Therefore, the use of AWM is essential to guarantee correctness.

\subsection*{8.3.2.2 Combination of AWM and Service Discovery}

In this section, we evaluate the combination of AWM and successful service discovery at runtime. In the depicted order-to-delivery process the vendors \((\text{XorVendor})\), delivery services \((\text{XorDelivery})\) and payment providers \((\text{XorPay})\) are bound at runtime. We investigate the influence of the number of discovered providers on our defined metrics. Thus, we vary the number of vendors \(i\), delivery services \(j\) and payment providers \(k\).

Obviously, if alternatives for services are integrated in the workflow in \(WP_{\text{XOR}}\) patterns, the success probability of the workflow increases. Numerous papers, e.g.,\([\text{SPJ09, Ste07, Ste08}]\) have evaluated the success probability in relation to produced costs. However, the effects of the integration of alternatives on workflow inherent conflicts has not

\(^{17}\text{http://www.paypal.de/de}\)
been addressed yet: The non-functional properties of bound services at runtime influence the properties of the $WP_{XOR}$ pattern they are element of. Thereby, conflicts may be avoided. Furthermore, if such an $WP_{XOR}$ pattern conflicts with another element, conflicts may be solved by eliminating alternatives, thus increasing the autonomy of services.

Our results are classified according to the defined metrics. We present results regarding the autonomy of elements and illustrate the influence of service discovery on semi-atomicity probability.

### Evaluating the Degree of Autonomy

**Test Set-Up** In this experiment we vary the number of discovered services $i, j$ and $k$ from $1, \ldots, 20$. We depict results for constant ratios of recoverable and redoable elements $p_{RC} = p_{RD} = 0.2, 0.5, 0.9$. The default for a fixed number of providers is 3.

**Assumption** When raising $i$ (while $j$ and $k$ remain constant), the number of recoverable elements in $XorVendor$ increases. Decisive for the conflict potential of $XorVendor$ and $XorDelivery$ (and $XorPay$ respectively) is the recoverability of $XorVendor$. If at least one recoverable element in $XorVendor$ exists, occurring conflicts are resolved by eliminating non-recoverable vendors. Thus, with increasing number of vendors $i$, we expect the autonomy to increase.

Additionally, if the number of delivery services $XorDelivery$ $j$ (or payment providers $k$ respectively) is raised, the chance that $XorDelivery$ becomes redoable increases. Thus, the conflict potential between $XorVendor$ and $XorDelivery$ is slightly decreased. However, the conflict potential between $XorVendor$ and $XorPay$ remains. Thus, in this case, we assume the degree of autonomy to moderately increase with increasing $j$.\footnote{We expect analog results for increasing $k$ while keeping $i$ and $j$ constant.}

If the number of delivery services $j$ and the number of payment providers $k$ are increased, the chance that the elements $XorDelivery$ and $XorPay$ are redoable, increases. Thereby, the conflict potential is reduced and $d_{AWM} (\omega)$ presumably ascends more quickly as opposed to the previous case.

**Evaluation** In Figure 8.12 on the left hand side, the results of the experiment increasing $i$ (and constant $j = k = 3$) are depicted. For great ratios $p_{RC}$ and $p_{RD}$ ($p_{RC} = p_{RD} = 0.9$), the autonomy is approximately 1 for all values $i$. Regarding $p_{RC} = p_{RD} = 0.2$, the resulting $d_{AWM} (\omega)$ is $d_{AWM} (\omega) \approx 0.83$ (for $i = 1$) and approaches $d_{AWM} (\omega) = 1$ (for $i \geq 14$). In all depicted cases, $d_{AWM}$ increases to 1. This meets our assumption: As more elements are discovered in the $XorVendor$ pattern, the number of recoverable vendors...
increases. If conflicts between \textit{XorVendor} and \textit{XorDelivery} or \textit{XorPay} exist, the non-recoverable vendors are eliminated to solve these conflicts.

![Figure 8.12: $d_{AWM}$ of the order-to-delivery process, varying the number of vendors $i$.](image1)

On the right side of Figure 8.12, the results for increasing the number of delivery providers $j$ (and constant $i = k = 3$) are provided.\textsuperscript{19} The process of the depicted results is similar to those depicted on the left hand side. However, as it can be seen for $p_{RC} = p_{RD} = 0.2$, the increase of $d_{AWM}$ is lower than on the left hand side. That is due to the fact, that the increase of delivery services does not the influence the remaining conflicts between \textit{XorVendor} and \textit{XorPay}. These conflicts remain and the autonomy does not increase as quickly as on the left hand side. In comparison to WS-AT, the degree of autonomy is raised by more than 80\% in this scenario.

![Figure 8.13: $d_{AWM}$ of the order-to-delivery process, varying the $j$ (delivery) and $k$ (pay).](image2)

In Figure 8.13, the degree of autonomy ($d_{AWM}(\omega)$ and $d_{AT}(\omega)$) for the order-to-delivery process is displayed, varying the number of delivery services $j$ and payment providers $k$ (with $j = k$). The trends of the depicted results are similar to those for varying $i$ (Figure 8.12, left hand side): For $p_{RC} = p_{RD} = 0.2$, $d_{AWM}$ increases from $d_{AWM}(\omega) \approx 0.82$ for $j = k = 1$ to $d_{AWM}(\omega) \approx 1$ for $i \geq 15$. Once again, our assumption is met, as in these

\textsuperscript{19}The results for increasing $k$ and keeping $i$ and $j$ fixed correspond to the illustrated results.
cases, the chance that \(\text{XorDelivery}\) and \(\text{XorPay}\) are redoable increases, thereby avoiding conflicts in the first place.

**Conclusion** These experiments confirm our assumption that the combination of AWM and successful service discovery positively influences the degree of autonomy achieved by AWM. Especially, if more vendors \(i\) are discovered or the number of delivery services \(j\) and payment providers \(k\) equally increases, \(d_{AWM}(\omega)\) quickly ascends. In the first case (raising \(i\)), conflicts are solved by eliminating non-recoverable alternatives. In the other case (increasing \(j = k\)), the chance of \(\text{XorDelivery}\) and \(\text{XorPay}\) being redoable is increased as one redoable alternative suffices to avoid conflicts in the first place. Overall, successful service discovery in terms of high hit ratios further increases the autonomy of services in adaptive workflow management.

**Evaluating the Semi-Atomicity Probability**

**Test Set-Up** Just as in the previous experiments, in this series of tests, we vary the number of bound vendors \(i\), delivery services \(j\) and payment providers \(k\) from 1, … 20. Again, the default value for fixed number of providers is 3.

**Assumption** As AWM guarantees semi-atomicity, we assume \(p_{SA}(\omega)_{AWM}\) to be 1 for all cases. Regarding the correctness guarantee for WS-BA, the decisive factor is the success probability of the \(WP_{XOR}\) patterns which increases with the number of alternatives. When raising the number of vendors \(i\) while keeping \(j\) and \(k\) fixed, the success probability of \(\text{XorVendor}\) increases, while the chance that it is recoverable remains fixed.\(^{20}\) However, the success probability of \(\text{XorDelivery}\) and \(\text{XorPay}\) remains constant. The chance for success of \(\text{XorVendor}\) in the presence of failure of \(\text{XorDelivery}\) or \(\text{XorPay}\) thereby increases. We thus assume \(p_{SA}(\omega)_{BA}\) to noticeably decrease.

If the number of delivery services \(j\) is raised, while keeping the number of vendors \(i\) and payment providers \(k\) constant, the success probability of \(\text{XorDelivery}\) increases. Thereby, the chance of \(\text{XorVendor}\) to complete while \(\text{XorDelivery}\) fails, decreases. We therefore assume \(p_{SA}(\omega)_{BA}\) to increase in this scenario.

If the number of delivery services \(j\) and payment providers \(k\) is raised while \(i\) remains fixed, the success of the latter \(WP_{XOR}\) patterns increases. We therefore assume the semi-atomicity probability \(p_{SA}(\omega)_{BA}\) to noticeably increase and reach 1.

**Evaluation** In Figure 8.14, \(p_{SA}(\omega)_{AWM}\) and \(p_{SA}(\omega)_{BA}\) are shown on the y-axis, while varying the number of vendors \(i\) on the x-axis. For WS-BA, the success probability of elements \((p_S = 0.5)\) as well as the ratio of recoverable elements \((p_{RC} = 0.5, 0.9)\)

\(^{20}\) In this case AWM eliminates inadequate branches, increasing the chance that \(\text{XorVendor}\) is recoverable.
remain fixed. Obviously, $p_{SA}(\omega)_{AWM} = 1$ for all values of $i$. For WS-BA, it can be easily seen that with increasing $i$ the chance of correct completion clearly decreases from approximately $p_{SA}(\omega)_{BA} \approx 0.9$ (for $i = 1$) to $p_{SA}(\omega)_{BA} \approx 0.4$ for $i = 20$. This is due to the fact that with ascending $i$ the chance that XORVendor completes increases while the chance for failures of XORDelivery or XORPay remains constant. As the probability that XORVendor is recoverable remains fixed, this increases the number of failures which cannot be recovered. Using AWM is therefore necessary to ensure correctness when employing service discovery to dynamically bind services.

Figure 8.14: $p_{SA}$ of the order-to-delivery process, varying the number of vendors $i$.

On the left hand side of Figure 8.15, $p_{SA}(\omega)_{BA}$ (and $p_{SA}(\omega)_{AWM}$) are presented on the y-axis while varying the number of bound delivery services $j$.\footnote{As the results for increasing $k$ expose the same conclusions, their presentation is spared.} Besides AWM, we depict results for half of the elements being recoverable, i.e., $p_{RC} = 0.5$ and different success probabilities $p_S = 0.6, 0.7$. It can be seen in this experiment that the resulting $p_{SA}(\omega)_{BA}$ slightly increases, however with perceptible deviation.

Figure 8.15: $p_{SA}$ of the order-to-delivery process, varying $j$ (delivery) and $k$ (pay).

On the right hand side of Figure 8.15, the results for increasing the number of delivery services and payment providers $j = k$ are depicted. While AWM guarantees correct
execution in all cases (i.e., \( p_{SA}(\omega)_{AWM} = 1 \)), the resulting \( p_{SA}(\omega)_{BA} \) using WS-BA increases considerably for both depicted scenarios (\( p_S = 0.5 \) and \( p_{RC} = 0.5, 0.9 \)). Thus, the chance for inconsistent system states decreases. As assumed, the chance of successful completion of \( XorDelivery \) and \( XorPay \) increases with ascending \( j = k \). Thereby, if half of the dynamically bound elements are recoverable (i.e., \( p_{RC} = 0.5 \)), the overall number of failures is decreased and \( p_{SA}(\omega)_{BA} \) increases from roughly \( p_{SA}(\omega)_{BA} \approx 0.42 \) for \( i = 1 \) to \( p_{SA}(\omega)_{BA} = 1 \) for \( i \geq 11 \).

**Conclusion** Evaluating the correctness guarantees produced by AWM in combination with service discovery, AWM ensures correct execution in all cases. Thereby, the forward-recovery potential is increased, thus increasing the overall chance of the workflow to successfully complete. The results of these tests reveal that depending on the position of the according \( WP_{XOR} \) in the workflow, the semi-atomicity probability employing WS-BA may be positively or negatively influenced. When integrating more vendors \( i \) while remaining the number of elements in \( XorDelivery \) and \( XorPay \) fixed, \( p_{SA}(\omega)_{BA} \) drastically decreases. However, if the number of alternatives of delivery services \( j \) as well as payment providers \( k \) is increased, the chance that the workflow semi-atomically completes using WS-BA increases. The alignment of the \( WP_{XOR} \) patterns in the workflow is decisive: \( XorVendor \) may only conflict with subsequent workflow elements, \( XorDelivery \) and \( XorPay \) only conflict with elements which are aligned prior to them.

**8.4 Summary**

In this chapter, we evaluated AWM according to the resulting autonomy of elements and ensured correctness guarantees and compared it to a pessimistic (WS-AT) and an optimistic (WS-BA) reference approach. WS-AT ensures correctness however does not grant autonomy to participants while WS-BA enables autonomous coupling however at the cost of correctness. Our approach AWM is a hybrid approach which ensures correct execution in any case and grants autonomy to participants whenever possible.

The evaluations reveal that the overall number of included elements is not decisive for the autonomy \( d_{AWM} \). However, it is strongly influenced by the transactional properties of services: Obviously, the more services are recoverable, the greater the resulting autonomy. In the extreme case, if all services are recoverable, AWM ensures autonomy of all services. In addition, the evaluation shows that the same holds for redoable elements: Independent of the ratio of recoverable elements, the more services are redoable, the greater the resulting \( d_{AWM} \).

Furthermore, data dependencies effect the autonomy as they increase the chance that services *directly transactionally conflict*. Apparently, the more elements are involved in data dependencies, the greater is the conflict potential and thus the lower the autonomy. However, the nature of data dependencies prominently influences the performance as
well: For constant ratios of data dependent elements, the resulting autonomy is greater
for short data dependency sequences than for longer ones. This is attributed to the
fact that for short data dependency sequences the overall number of dependencies is
lower than for long sequences. In addition, in case of longer data dependency sequences,
AWM benefits from sparing the coordination of indirect conflict elements. These are
not considered by WS-AT or WS-BA.

As shown in Section 6.3.4, AWM ensures correct execution in all cases, just as the pes-
simistic approach WS-AT. Employing the optimistic approach WS-BA, the correctness
of the execution is jeopardized as soon as one non-recoverable element is included. Gen-
erally, the less services are recoverable, the more likely it is that the process will not semi-
atomically terminate. However, in sequential alignments, the guarantees strongly differ
according to the position of non-recoverable elements: The earlier a non-recoverable
element is invoked in the process, the more likely it will not semi-atomically terminate.

Further factors which influence the correctness of WS-BA, \( p_{SA}(\omega)_{BA} \), are the size of the
workflow as well as the success probability of included services: \( p_{SA}(\omega)_{BA} \) is determined
by the chance for successful execution of a workflow \( \omega \) as well as the probability of
recoverable failures. Thus, for low success probabilities of services \( p_s \), the chance for
recoverable failures is high (cf. if the first invoked element fails). When increasing \( p_s \),
\( p_{SA}(\omega)_{BA} \) decreases. If \( p_s \) is further increased, the chance for successful termination of
\( \omega \) increases, thus \( p_{SA}(\omega)_{BA} \) again increases.

Especially, when integrating service discovery to enable forward-recovery in case of
failure, the evaluation conveys that the use of AWM is essential: Depending on the
position of the alternatives in \( \omega \), \( p_{SA}(\omega)_{BA} \) might even decrease with increasing number
of alternatives. AWM however ensures correct execution while further increasing the
degree of autonomy in all cases.

Overall, the autonomy given by AWM is greater than the autonomy of WS-AT in
almost all cases. Only in the ‘worst case’, if none of the elements is recoverable nor
redoable, they are equal, i.e., \( d_{AWM} = d_{AT} \). Additionally, correctness guarantees given
by AWM are always 100%, thus greater than or equal to that of WS-BA. If only 1%
of the services are non-recoverable, \( p_{SA}(\omega)_{BA} \) is decreased to only 50% in certain set-
ups. As furthermore, WS-BA does not offer any possibility to validate a given process,
the deployment of AWM is essential to ensure correct execution in the sense of semi-
atomicity. We are able to validate our experimental results by our analytical approaches
for \( d_{AWM}(\omega) \) and \( p_{SA}(\omega)_{BA} \).
9 Conclusion

This thesis has been dedicated to develop transactional support for ad-hoc collaborations in dynamic environments. These are specified as composite services and implemented as workflows. Service discovery enables dynamic binding of components at runtime. We introduced a formal model of ad-hoc collaborations and based on this, presented our approach of adaptive workflow management. It employs verification and adaptation algorithms to ensure correct execution of workflows: A workflow is adapted prior to runtime if its verification fails, and additionally dynamically during execution in case of failure of elements or discovery of alternatives.

Introducing this, we contributed a novel transactional abstraction layer to existing workflow management systems: A designer is no longer compelled to statically defining workflows along with their complete failure handling at design time. Instead, she is provided with means of specification of abstract workflows at design time; failure handling for these is automated at runtime according to dynamically discovered components. Correctness of specific executions of these workflows at runtime is guaranteed. Thereby, a new degree of flexibility for workflows is achieved. The main contributions of this thesis are summarized as follows:

Transactional Support of Ad-hoc Collaboration  Especially when implementing ad-hoc collaborations in mobile environments, the ability to flexibly discover and compose services at runtime is indispensable: In such dynamic environments, the execution context, i.e., available services at runtime, might not even be known at design time and might differ from execution to execution. Our approach integrates service discovery, verification, and dynamic composition of workflows at runtime to ensure reliable cooperations: A composition is altered and automatically enhanced by appropriate failure handling mechanisms to ensure correct execution of these composite services in the presence of failures. Failure recovery mechanisms are dynamically identified according to the requirements of the ad-hoc cooperations and its participants.

Autonomy vs. Reliability  Our approach to guarantee transactionally correct execution of workflows in the sense of failure atomicity, abandons from tight coupling of services to transactions if possible, thus granting autonomy to participants. However, enabling autonomous execution of components comes at the cost of relaxing strict correctness
criteria, such as strict atomicity, e.g., as ensured by 2PC. We introduced semi-atomicity as the correctness criterion for composite services in the presence of transactional properties. It allows for completion of services at different times and exploits capabilities to ensure consistent system states by employing convenient forward- or backward-recovery. Conventional approaches so far either strictly bind the execution of participating services to transaction phases (thus ensure correctness however do not grant autonomy) or they loosely couple services to activities while assuming compensatability of all involved entities. As our evaluation showed, correctness of such workflows is already jeopardized as soon as one element is not compensatable. As opposed to these, we presented a hybrid approach that ensures reliable, i.e., correct ad-hoc cooperation in any case, and grants autonomy to participants whenever possible.

**Minimal Set of Coordinated Elements** Employing our formal model, we are able to identify the minimal set of elements which need to be coordinated. We are thereby able to identify the minimal set of nodes, whose autonomy needs to be limited in order to guarantee correctness in any case. On the other hand, we proved the minimality property of this set: i.e., no more than these elements need to be coordinated in order to ensure semi-atomicity. Therefore, we were able to show that our algorithm to adapt workflows at runtime, outputs optimal results regarding the number of coordinated elements and thus the autonomy of participants.

**Forward-Recovery by Service Discovery** We presented a service discovery protocol for mobile ad-hoc networks which exploits the mobility of participants to dynamically find and make use of services. We identified service discovery as a prerequisite to enable ad-hoc collaborations, as services may only be dynamically composed if they are discovered. Additionally, by integrating service discovery and adaptive workflow management, we are able to incorporate alternatives for services at runtime thereby increase the overall chance for successful termination. Furthermore, we demonstrated that by doing so, diversity of transactional properties resolves conflicts and enables exclusion of transactionally conflicting elements. All in all, we showed that service discovery enables forward-recovery of ad-hoc cooperations at runtime thus increasing the overall chance for successful termination as well as the autonomy of participants.

We introduced our adaptive workflow management system and gave a formal demonstration of its correctness. Furthermore, we presented a prototypical implementation which realizes our approach. The results of our experimental evaluation show that our hybrid approach is very well suitable in a variety of system settings. We verified all experimental results employing our analytical approach.

Our presented work builds the theoretical foundation for ensuring transactional cor-
rectness of flexible workflow systems. Existing approaches focus on optimization of quality of service aspects, such as utility costs, response time, or information quality. Complementing these with our approach is worthwhile investigating in future work, as it allows for the examination of trade-offs between autonomy of components and individually determined optimization of costs and benefits despite transactional correctness. Furthermore, an approach which probabilistically determines the transactional properties of patterns according to the number and properties of its alternatives is conceivable: Thereby, further flexibility, optimization potential, and autonomy of services may be obtained - whereas the correctness guarantees will be given probabilistically as well.
Bibliography


Bibliography


Bibliography


A Formal Model

A.1 Basic Workflow Patterns

The following workflow patterns are defined by the WfMC as the basic workflow patterns.

- A sequence of services denotes, that one service is enabled after the completion of the preceding task. This is also known as sequential or serial routing.

- The parallel split of a branch leads to a decomposition of one branch into several which are all executed in parallel. Different denotations of this pattern are AND-split, parallel routing or fork.

- At the synchronization point, several branches are converged into one single subsequent branch. The subsequent branch is enabled as soon as all incoming branches completed. This pattern is referred to as AND-join, rendezvous or synchronizer.

- The exclusive choice in a workflow denotes the divergence of a single branch into several branches out of which one and only one branch is enabled. Synonyms for the exclusive choice are XOR- or exclusive OR-split, conditional routing, switch, decision or case statement.

- The simple merge converges several incoming branches to a single outgoing branch. The subsequent workflow is enabled every time an incoming branch completes. This pattern is also known as XOR-join, exclusive OR-join, asynchronous join or merge.

A.2 Advanced Workflow Patterns

The following advanced workflow patterns (as specified by the WfMC) can be reduced to the introduced patterns. Therefore, we disregard them in the scope of this thesis:

- Multi choice (OR-split) and multi synchronization respectively (OR-join) denote a pattern, in which one or several of the branches are executed. The subsequent workflow is activated as soon as all activated branches within the pattern are completed. This pattern can be reconstructed using $WP_{XOR}$. For example, the multi-choice pattern consisting of two service $s_1$ and $s_2$ (i.e., $OR(s_1, s_2)$) is equivalent to the following representation using the $WP_{XOR}$ pattern: $WP_{XOR}(s_1, s_2, WP_{AND}(s_1, s_2))$. 
The n-out-of-m pattern as originally defined states that n or more branches out of m possible branches have to complete in order to successfully complete the pattern. Although more complex, this again can be constructed by WP\(^{\text{XOR}}\) pattern. Consider for example an n-out-of-m pattern with n = 2 out of the m = 3 branches \((s_1, s_2, s_3)\). This is equivalent to the following pattern:

\[ WP^{\text{XOR}}(WP^{\text{AND}}(s_1, s_2), WP^{\text{AND}}(s_1, s_3), WP^{\text{AND}}(s_2, s_3), WP^{\text{AND}}(s_1, s_2, s_3)) \]

Other advanced patterns (multi-merge, structured discriminator, etc.) comprise aspects of workflows which are different from transactional execution. Especially, there behavior is not deterministic. According to their definition, it cannot be reconstructed in case of completion, whether one or several services completed and whether they have been executed one or several times respectively. We therefore disregard these in our work.

### A.3 Complete Transactional Properties of Patterns

The complete transactional properties of \( WP^{\text{SEQ}}(E) \) and \( WP^{\text{AND}}(E) \) pattern are defined as follows:

**Definition 24. Complete Transactional Properties of \( WP^{\text{SEQ}}(E) \) and \( WP^{\text{AND}}(E) \)**

A pattern \( WP(E) \), containing of the set of elements \( E \), which is a Sequence \( WP^{\text{SEQ}}(E) \) or an And \( WP^{\text{AND}}(E) \) pattern,

- is *compensatable*, if and only if all included elements are compensatable
  \[ WP(E).\text{compensatable} = 1 \]
  \[ \iff \forall e \in E : e.\text{compensatable} = 1 \]

- needs *consistent completion*, as soon as one element needs consistent completion
  \[ WP(E).\text{consistentCompletion} = 1 \]
  \[ \iff \exists e \in E : e.\text{consistentCompletion} = 1 \]

- is *retrieable*, if all elements are retrieable
  \[ WP(E).\text{retrieable} = 1 \]
  \[ \iff \forall e \in E : e.\text{retrieable} = 1 \]

The complete transactional properties of \( WP^{\text{XOR}}(E) \) patterns are defined as follows.
Definition 25. Transactional Properties of $WP_{\text{XOR}}(E)$

An XOR pattern $WP_{\text{XOR}}(E)$ containing the set of elements $E$

- **is compensatable**, if all enclosed elements are compensatable. It is **not compensatable**, if none of the enclosed elements is compensatable. Otherwise, it is not known:
  \[
  WP_{\text{XOR}}(E).\text{compensatable} = 1
  \quad \iff \forall e \in E : e.\text{compensatable} = 1
  
  WP_{\text{XOR}}(E).\text{compensatable} = 0
  \quad \iff \forall e \in E : e.\text{compensatable} = 0
  \]

- **needs consistent completion**, if all included elements demand consistent completion. If none of the included elements demands consistent completion, the pattern allows for inconsistent completion. Otherwise, it is not known:
  \[
  WP_{\text{XOR}}(E).\text{consistentCompletion} = 1
  \quad \iff \forall e \in E : e.\text{consistentCompletion} = 1
  
  WP_{\text{XOR}}(E).\text{consistentCompletion} = 0
  \quad \iff \forall e \in E : e.\text{consistentCompletion} = 0
  \]

- **is retrieable**, as soon as one element is retrieable. Otherwise, it is not retrieable.
  \[
  WP_{\text{XOR}}(E).\text{retrieable} = 1
  \quad \iff \exists e \in E : e.\text{retrieable} = 1
  \]
B Adaptive Workflow Management

B.1 Proof of Proposition 5

Proof. • ‘⇒’: Assume, no transactional conflicts \{e_i, e_j\}_C with e_i, e_j \in E exist. Consider e ∈ E with pt(e) ≠ (1, 1). Distinction of cases:

- If pt(e) = (0, 0): Assume it exists e′ ∈ E, with e ≠ e′ and pt(e′) ≠ (1, 1), then \{e, e′\}_C transactionally conflict. Contradiction to assumption, thus for all e′ ∈ E, with e ≠ e′ it holds: pt(e′) = (1, 1). This fulfills condition c.

- Else if pt(e) = (1, 0): Assume it exists e′ ∈ E, with e ≠ e′ and pt(e′) ≠ (1, *), then \{e, e′\}_C transactionally conflict. Contradiction to assumption, thus ∀e′ ∈ E, with e ≠ e′ it holds: pt(e′) = (1, *). According to Definition 24, WP_{AND}(E) is then recoverable pt(WP_{AND}(E)) = (1, 1). This fulfills condition a.

- Else if pt(e) = (0, 1) (analogue): Assume it exists e′ ∈ E, with e ≠ e′ and pt(e′) ≠ (*, 1), then \{e, e′\}_C transactionally conflict. Contradiction to assumption, thus ∀e′ ∈ E, with e ≠ e′ it holds: pt(e′) = (*, 1). According to Definition 24, WP_{AND}(E) is then retrieable pt(WP_{AND}(E)) = (*, 1). This fulfills condition b.

Else if ∀e ∈ E it holds pt(e) = (1, 1), according to Definition 24, pt(WP_{AND}(E)) = (1, 1), thus condition a, b and c are fulfilled.

• ‘⇐’: Assume, that WP_{AND}(E) is recoverable, retrievable or condition c is fulfilled. Consider the following distinction of cases:

- pt(WP_{AND}(E)) = (1, *): Thus, all e ∈ E are recoverable (pt(e) = (1, *)) and therefore no conflict exists.

- pt(WP_{AND}(E)) = (1, *) accordingly: Thus, all e ∈ E are retrieable (pt(e) = (*, 1)) and therefore no conflict exists.

- Condition c is fulfilled: Let e ∈ E be the element with pt(e) = (0, 0). As all other elements e′ ∈ E employ pt(e) = (1, 1), no conflicts exist.

\qed
B.2 ATS-Invariance of Resulting $\omega'$

As defined in Section 6.3.2, the adaptations are correct, if $ATS_{\omega'} \Rightarrow ATS_{\omega}$.

The adaptation algorithm operates on the data dependency graph $G_\omega(V, E)$. The set of nodes $V$ contains all mandatory elements of $\omega$. The completion of $\omega$ is ensured, if all $v_i \in V$ complete. Thus, the ATS-representation of $\omega$ is:

$$ATS_\omega = \bigwedge_{v \in V} ATS_{vi}$$

Elements of $V$ are either services or $WP_{XOR}$ patterns. Let $V_x \subseteq V$ be the set of $WP_{XOR}$ patterns in $\omega$. Let additionally $\{x_{i,j}\}_{j=1}^{n}$ be the set of alternatives in $v_i \in V_X$. The ATS-representation of an element $v_i \in V$ is then:

$$ATS_{vi} = \begin{cases} \bigvee_{j=1}^{n} (x_{i,j} \land ( \bigwedge_{k=1}^{n} \neg x_{i,k}) & \text{if } v_i \text{ is a } WP_{XOR} \\ v_i & \text{else} \end{cases}$$

Note, that $ATS_v$ for $WP_{XOR}$ patterns consists of a mutually exclusive disjunction of conjunctions. That implies, at most one of the conjunctions $x_{i,j} \land ( \bigwedge_{k=1}^{n} \neg x_{i,k})$ is true. Using this, the ATS-representation of $\omega$ is:

$$ATS_\omega = ( \bigwedge_{v \in V \setminus V_x} v) \land ( \bigwedge_{v \in V_x} ATS_{vi})$$

The adaptation algorithm (Algorithm 2) traverses $G_\omega(V, E)$ and processes every node $v \in V$. The data dependency graph of $G_\omega$ is thus equivalent to $G_{\omega'}$. However, the $WP_{XOR}$ nodes might be altered in the second step by eliminating branches.

Let $v_i = WP_{XOR}(\{x_{i,j}\}_j)$ be altered to $v'_i = WP_{XOR}(\{x_{i,j}\}_j \setminus \{x_{i,l}\}_l)$ by eliminating all branches $\{x_{i,l}\}_l$. For convenience, let $L$ be the set of indices of the above eliminated branches. The ATS-representation of the $WP_{XOR}$ is thereby altered to:

$$ATS_{v_i} = \bigvee_{j=1}^{n} (x_{i,j} \land ( \bigwedge_{k=1}^{n} \neg x_{i,k}))$$

Obviously, as at most one of the conjunction terms $x_{i,j} \land ( \bigwedge_{k=1}^{n} \neg x_{i,k})$ is true, it holds: $ATS_{v_i} \Rightarrow ATS_{v'_i}$.

The ATS-representation of $\omega'$ is therefore:

$$ATS_{\omega'} = ( \bigwedge_{v \in V \setminus V_x} v) \land ( \bigwedge_{v'_i \in V_x} ATS_{v'_i})$$

1For the sake of readability, we abandon duplicate subscripts. Thus, we utilize $ATS_{vi}$ instead of $ATS_{v_i}$. 
B.2 ATS-Invariance of Resulting $\omega'$

As accepted termination of the altered $WP_{XOR}$ patterns $ATS_{\omega'}$ implicate accepted termination of the original patterns $ATS_{\omega}$, it also holds that:

$$ATS_{\omega'} \Rightarrow ATS_{\omega}$$

We thereby demonstrated, that the output $\omega'$ is ATS-invariant to the input workflow $\omega$. 
The implementation of a \( WP_{XOR}(S_1, S_2) \) by AWM is shown in Figure C.1.

Figure C.1: Implementation of the \( WP_{XOR} \) pattern.
D Evaluation

D.1 Expected Index $c_0$ of the First Non-Recoverable Element

When given the number $n$ of elements within a sequence, i.e., $WP_{SEQ}(s_1, \ldots, s_n)$, and the number of non-recoverable elements $c$, the index of the first non-recoverable element $c_0$ is approximated as follows. Let $\{s_{c_0}, \ldots s_{c}\}$ denote the sequence of non-recoverable elements.

As $c$ and $n$ are given, it obviously holds: $c_0 \in \{1, \ldots n - (c - 1)\}$.

If $c_0 = n - (c - 1)$, there exists exactly one possible alignment for the non-recoverable elements. All recoverable elements are aligned prior to $c_0$, all non-recoverable elements behind $c_0$.

If $c_0 = n - (c - 1) - 1 = n - c$, one recoverable and $c - 1$ non-recoverable elements follow $c_0$ (i.e., overall $c$ elements succeed $s_{c_0}$). Thus, there are $\binom{c}{c-1}$ possible alignments for the recoverable element behind $c_0$.

If $c_0 = n - c - 1$, 2 recoverable and $c - 1$ non-recoverable elements follow $c_0$ (i.e., overall $c + 1$ elements succeed $s_{c_0}$). Thus, there are $\binom{c+1}{c-1}$ possible alignments for the non-recoverable elements behind $c_0$.

Regarding all possible values of $c_0 \in \{1, \ldots n - c + 1\}$, the expected value for $c_0$ is therefore determined as follows:

$$E(c_0 = i) = \frac{\sum_{i=1}^{n-c+1} i \cdot \binom{n-i}{c-1}}{\sum_{i=1}^{n-c+1} \binom{n-i}{c-1}}$$

D.2 Evaluating $d_{AWM}$ Varying the Number of Included Services $n$

Test Set-Up In this series of tests, we vary the number of elements $n$ within the workflow from $n = 1, \ldots, 100$ and present the results for different settings with normally distributed ratios of transactional properties around a stated mean (variance 5%). This mean is chosen to be equal i.e., $p_{RC} = p_{RD}$ for all tests. We perform the analysis for test scenarios without and with given data dependencies. We present the results for the following types of workflows:
D.2 Evaluating $d_{AWM}$ Varying the Number of Included Services $n$

- $\omega'$ is a workflow without data dependencies, consisting of $n$ services, thus:
  $\omega' = WP_{AND}(s_1, \ldots, s_n)$

- $\omega''$ contains data dependencies of the following form: Half of the included elements, that is $n/2$ elements, are randomly chosen to be sequentially data dependent, e.g., $s_1 \rightarrow \ldots \rightarrow s_{n/2}$. All other $n/2$ elements are randomly aligned in parallel or sequence. Thus, one possible alignment of $\omega''$ is then:
  $\omega'' = WP_{AND}(WP_{SEQ}(s_1, \ldots, s_{n/2}), s_{n/2+1}, \ldots, s_n)$.

- $\omega'''$ contains $t$ data dependencies (with $t < n$), each consisting of exactly 2 elements, e.g. $s_i \rightarrow s_{i+1}$. All other $n - 2 \times t$ elements are randomly appended in sequence or parallel. Thus, one characteristic alignment of $\omega'''$ is:
  $\omega''' = WP_{AND}(WP_{SEQ}(s_1, s_2), \ldots WP_{SEQ}(s_{2t-1}, s_{2t}), s_{2t+1}, \ldots, s_n)$.

According to the varied parameters, the input workflows are generated, transformed and analyzed. The given results reflect average values of at least 100 test runs.

Assumption Recall the analytical approach for determining the degree of autonomy of a workflow with and without data dependencies. Without data dependencies, i.e., $\omega'$, it is straightforward to see, that $d_{AWM}$ is independent of $n$. Therefore, we assume $d_{AWM}(\omega')$ to converge to a constant value.

When varying the data dependencies with respect to the size of the workflow $n$, $d_{AWM}$ is presumably equal to or greater than $d_{AT}$ by a constant $c_a$. Therefore, our assumption for this series of tests is formulated as follows:

- $d_{AWM}$ is (nearly) constant when varying $n$, thus $d_{AWM}(\omega) \rightarrow d_{AT}(\omega) + c_a$.

Evaluation In Figure D.1 the degree of autonomy for AWM $d_{AWM}(\omega')$ and for WS-AT $d_{AT}(\omega')$ is illustrated for the example workflow $\omega'$ (i.e., without data dependencies). The size $n$ of the workflow $\omega'$ is varied from $n = 1, \ldots, 100$ (on the x-axis). Besides $d_{AT}(\omega')$, $d_{AWM}(\omega')$ is depicted for different ratios of transactional properties $p = 0.2$ and $p = 0.5$ (with $p = p_{RC}(\omega) = p_{RD}(\omega)$).

For $n = 1$ all degrees are 1, as transactional coordination of a single service is not necessary. For $n \geq 2$, $d_{AWM}(\omega')$ (for all values of $p$) is constantly greater than $d_{AT}(\omega')$, thus fewer elements have to be coordinated. The degree of autonomy for WS-AT $d_{AT}(\omega')$ is 0, as all elements are coordinated. Consider $d_{AWM}(\omega')$ for $p = 0.2$: The resulting $d_{AWM}(\omega')$ converges to $d_{AWM}(\omega') = 0.36$. This is verified, applying Formula 8.2: $d_{AWM}(\omega') = 1 - (1 - 0.2) \times (1 - 0.2) = 1 - 0.64 = 0.36$, thus $0.64 \times n$ elements are coordinated in $\omega'$. This is analogue for $p = 0.5$ (resulting in $d_{AWM}(\omega') = 0.25$).

In Figure D.2, we depicted the results for $\omega''$ (on the left hand side) and $\omega'''$ (on the right hand side), thus for workflows with data dependencies. In both depicted results, half of the elements are recoverable and half of them redoable, thus $p_{RC} = p_{RD} = 0.5$. 

143
D.2 Evaluating $d_{AWM}$ Varying the Number of Included Services $n$

![Figure D.1](image1.png)

*Figure D.1: $d_{AWM}$ of $\omega'$ without data dependencies, varying $n$."

![Figure D.2](image2.png)

*Figure D.2: $d_{AWM}$ of $\omega''$ (left) and $\omega'''$ (right) varying $n$."

In $\omega''$ a data dependency sequence of half of the size of the workflow $n/2$ is included. Thus, indirect conflict elements may occur. In Figure D.2 on the left hand side, we depicted the degree of autonomy of AWM and additionally the degree of autonomy, if indirect conflict elements are not excluded from coordination (AWM'). For small values of $n$, $d_{AWM}(\omega'')$ (and $d_{AWM}'(\omega'')$) range around 0.75 (which is the limit for $p = 0.5$ without data dependencies in Figure D.1). Both curves decrease and converge to a stable value (approximately $d_{AWM}(\omega'') = 0.5$ and $d_{AWM}'(\omega'') = 0.4$). This is confirmed by our analytical approximations, using Formulas 8.3 and D.1 (see Appendix D.3).

In Figure D.2 on the right hand side, the evaluation results for workflows of type $\omega'''$ in which $t = 0.2 \ast n$ (i.e., $r = 0.2$) elements are data dependent on one other element are presented. I.e., if 10 elements are included in the workflow, 2 data dependencies exist.$^1$

At the beginning ($n \leq 5$), $d_{AWM}(\omega''')$ varies between roughly $d_{AWM}(\omega''') = 0.8$ and $d_{AWM}(\omega''') = 0.75$. For greater values of $n$ it converges to $d_{AWM}(\omega''') = 0.75$. This again is confirmed by our analytical results, using Formulas 8.3 and 8.4 (see Appendix D.3).

The results verify our assumption, that the autonomy of $\omega$ is not directly dependent on the size of the workflow. The variations in this experiment (i.e., neither $d_{AWM}(\omega)$ nor

---

$^1$In this scenario, no indirect conflict elements may occur, thus AWM' is not depicted.
Conclusion  In this series of tests, we inspected the influence of the size of the workflow $n$ on the degree of autonomy of AWM and the reference approach WS-AT. We were able to validate our assumption, that $d_{AWM}$ is constantly greater than $d_{AT}$. In the presented scenarios, the autonomy enabled by AWM $d_{AWM}$ is up to 75% greater than for WS-AT $d_{AT}$. The experimental results meet our results achieved by the analytical approach. The size $n$ of the workflow $\omega$ does not effect the resulting degree of autonomy $d_{AWM}(\omega)$. We therefore forego to present results for different values of $n$ when examining the degree of autonomy.

D.3 Analytical Approximation of $d_{AWM}$

Analytical Approximation of $d_{AWM}$ Varying $n$

In order to approximate the expected size of $M$ for $\omega''$, $s'_{AWM(\omega'')}$, we regard two parts of the workflow separately: The first one contains all elements, which are not involved in any data dependency (thus randomly aligned). The number of conflict elements among them is approximated employing Formula 8.2 on page 94.

The second part of the workflow consists of the elements which are data dependent on their predecessor. The number of conflict elements in this sequence is estimated using Formula 8.4 on page 95.

The union of the mentioned two sets of conflict elements yields to the approximation of the size of $M$ for $\omega''$. If $\omega''$ consists of $n$ elements, the expected size of $M$ is the sum of the conflict elements of the just mentioned parts of the workflow:

$$s'_{AWM}(\omega'') = (1 - p_{RC}(\omega_{n/2})) \cdot (1 - p_{RD}(\omega_{n/2})) \cdot n/2 + s'((\omega_{n/2}-1) \cdot p_{RD}(\omega_{n/2})$$

$$+ \sum_{i=2}^{n/2} i \cdot p_{RC}(\omega_{n/2})^{n/2-i} \cdot (1 - p_{RC}(\omega_{n/2})) \cdot (1 - p_{RD}(\omega_{n/2}))$$

We employ this to approximate the expected degree of autonomy $d'_{AWM}(\omega'')$, for fixed ratios of recoverable and redoable elements ($p_{RC}(\omega'') = p_{RD}(\omega'') = 0.5$) and varied the number of elements from $n = 1, \ldots, 100$. The results are depicted in Figure D.3. As it can be seen, the results achieved employing the analytical approach matches the degree of autonomy $d'_{AWM}(\omega'')$ achieved by the experiments. We validated this for different ratios of $p_{RC}(\omega'')$ and $p_{RD}(\omega'')$. 

$\omega''$ are constant) are due to the varying size of the sequence of data dependent elements in relation to $n$, which are further investigated in Section 8.2.1.3.
D.4 Evaluating $p_{SA}$ Varying the Number of Included Services $n$

Figure D.3: $d_{AWM}$ of $\omega''$ (left) and $\omega'''$ (right), compared to the analytical approximation.

**Analytical Approximation of $d_{AWM}$ Varying $p_{RD}$**

In Figure D.4, the resulting degree of autonomy disregarding indirect conflict elements $d'_{AWM}(\omega'')$ of workflow $\omega''$ are depicted. In addition to the experimental results, the analytical approximation of $d'_{AWM}(\omega'')$ (depicted as dashed lines) are illustrated. The analytical results are achieved by Formula D.1 in Appendix D.3. As it can be seen, the experimental results match are verified by the analytical determination of $d'_{AWM}(\omega'')$.

Figure D.4: $d_{AWM}$ of $\omega''$ varying $p_{RD}$ compared to the analytical approximations.

In Figure D.4, the resulting degree of autonomy disregarding indirect conflict elements $d'_{AWM}(\omega''')$ of workflow $\omega'''$ are depicted. In addition to the experimental results, the analytical approximation of $d'_{AWM}(\omega''')$ (dashed lines) are illustrated. The analytical results are again achieved by Formula D.1. As it can be seen, the experimental results match the analytical determination of $d'_{AWM}(\omega''')$.

D.4 Evaluating $p_{SA}$ Varying the Number of Included Services $n$

**Test Set-Up**  In this series of tests, we vary the number of elements $n$ from $n = 2, \ldots, 100$. We present the results for parallel ($WP_{AND}$) and sequentially ($WP_{SEQ}$) aligned services.
for vicarious values for the ratio of recoverable services \( p_{RC}(\omega) \) and the success probability \( p_S \).

**Assumption**
Recall, that through the adaptation, AWM guarantees semi-atomicity in all cases, thus \( p_{SA}(\omega)_{AWM} = 1 \) for all workflows \( \omega \). For the optimistic approach WS-BA, the following holds: With ascending size of workflows \( n \) (with a fixed ratio of recoverable elements \( p_{RC}(\omega) \)), the absolute number of non-recoverable elements increases. Thus, if the elements are arranged in parallel, the chance that one of the non-recoverable elements successfully completes while any other element fails, increases. We therefore assume \( p_{SA}(\omega)_{BA} \) to decrease with increasing \( n \), thus the optimization potential which AWM exploits to increase.

On the other hand, if elements are aligned in sequence, the influence of \( n \) according to the analytical model is more sophisticated: Obviously, for \( n = 1 \), \( p_{SA}(\omega)_{BA} = 1 \). With ascending size \( n \), the expectation value for the index of the first non-recoverable element \( c_0 \) increases as well. The exact value is dependent on \( n \) and the ratio of recoverable elements \( p_{RC} \).\(^2\) In case of failure, the semi-atomicity is ensured, if the failure occurs before the \( c_0 \)th element. For small values of \( c_0 \), the chance for such a failure is small and further decreases with ascending \( n \). For greater values of \( c_0 \), the chance for failures before the \( c_0 \)th element increases. Thus, we assume, that for greater values of \( p_{RC}(\omega) \), \( p_{SA} \) converges to 1 when increasing the size of the workflow \( n \). In conclusion that is, regarding sequential alignments for elements, \( p_{SA} \) is not mainly determined by \( n \), rather than \( p_S \) and \( p_{RC} \).

**Evaluation**
In Figure D.6, the \( p_{SA}(\omega)_{AWM} \) and \( p_{SA}(\omega)_{BA} \) are depicted (on the y-axis) for \( p_S = 0.9, p_{RC} = 0.1 \) and \( p_S = 0.5, p_{RC} = 0.99 \). The results of the parallel alignments \( WP_{AND} \) are shown on the left side, the results of sequential alignments \( WP_{SEQ} \), are

\(^2\)See Appendix D.1.
illustrated on the right side. The size of the workflow is varied on the x-axis. As assumed, \( p_{SA}(\omega)_{AWM} \) is 1 for all values of \( n \) in both scenarios.

Figure D.6: \( p_{SA} \) of an \( WP_{AND} \) and \( WP_{SEQ} \) pattern varying the number of elements \( n \).

Regarding the \( WP_{AND} \) pattern, it easily becomes apparent, that the resulting \( p_{SA}(\omega)_{BA} \) values for both depicted set-ups decrease with increasing \( n \). In the case of the relatively low ratio of recoverable elements \( p_{RC} = 0.1 \), \( p_{SA}(\omega)_{BA} \) quickly approaches zero. For the fairly great ratio of recoverable elements \( p_{RC} = 0.99 \), \( p_{SA}(\omega)_{BA} \) continuously decreases to values slightly lower than 0.6 for \( n = 100 \). In this case, due to the extremely high ratio of recoverable elements, the chance for recoverable failures (i.e., concurrent failures of all of the \((1 - p_{RC}) \times n\) non-recoverable elements) is still considerably high for great values of \( n \). However, vice versa speaking: Already a ratio of 1% non-recoverable elements (thus one non-recoverable element for \( n = 100 \)), clearly decreases the probability for \( \omega \) to correctly finish to lower than 0.6.

Using \( p_{RC} \) and \( n \) to determine the number of non-recoverable elements, the experimental results are verified applying Formula 8.7 on page 97: If the success probability of services is rather high \( p_{S} = 0.9 \), and the ratio of recoverable elements is \( p_{RC} = 0.1 \) the chance that \( n = 20 \) concurrently executed services semi-atomically terminate is: \( 0.9^{20} + 0.1^2 \approx 0.13 \).

On the right hand side of Figure D.6, the resulting \( p_{SA} \) for the same series input parameters (i.e., \( p_{S} = 0.9 \), \( p_{RC} = 0.1 \) and \( p_{S} = 0.5 \), \( p_{RC} = 0.99 \)) in sequential alignment \( WP_{SEQ} \) are depicted. As assumed, \( p_{SA}(\omega)_{AWM} \) equals 100% in any case.

Considering the resulting \( p_{SA}(\omega)_{BA} \), our assumption, that \( p_{SA}(\omega)_{BA} \) in this alignment is not solely dependent on \( n \) (rather than \( p_{RC} \) and \( p_{S} \)) is confirmed. For low ratios of recoverable elements \( p_{RC} = 0.1 \), \( p_{SA}(\omega)_{BA} \) continuously decreases with ascending \( n \). Using \( n \) and \( p_{RC} \) to approximate the expectation value for the index of the first non-recoverable element \( c_{0} \), we are able to verify this result applying Formula 8.8. E.g., for \( n = 100, p_{RC} = 0.1 \), the expectation value for \( c_{0} \) equals \( c_{0} = 1.1 \), thus resulting in \( p_{SA}(\omega)_{BA} \approx 0.1 \).
Regarding the second depicted series of tests with many recoverable elements (i.e., $p_{RC} = 0.99$), the $p_{SA}(\omega)_{BA}$ decreases to its minimum of roughly $p_{SA}(\omega)_{BA} \approx 0.5$ at $n \approx 15$. After that, $p_{SA}(\omega)_{BA}$ increases and quickly approaches 1. This behavior of $p_{SA}(\omega)_{BA}$ is explained as follows: For low values of $n$, the success of the whole workflow (quantified by the first term in Formula 8.8: $p^n$) is decisive. However with increasing $n$, this term approaches 0. On the other hand, the probability for semi-atomic termination in case of failure (quantified by the second term in Formula 8.8: $1 - p_S^k$) is roughly 0 for small values of $n$ and noticeably increases with ascending $n$, starting from $n \approx 15$. This is due to the fact, that for greater values of $n$ (and thus $c_0$, e.g. for $n = 100$, $p_{RC} = 0.99$, $c_0 \approx 27$) the probability, that failure occurs before the $c_0$th element approaches 1. Employing the analytical approach in Formula 8.8, we are able to verify the experimental results.

**Conclusion** In all performed experiments, our claim that $p_{SA}(\omega)_{AWM} \geq p_{SA}(\omega)_{BA}$ holds. Generally, in parallel alignments $WP_{AND}$, with increasing $n$, $p_{SA}(\omega)_{BA}$ decreases. However, the nature of the decrease (gradually or asymptotically) greatly differs depending on $p_{RC}$ and $p_S$. Our experiments prove, that even if the ratio of non-recoverable elements is very low (i.e., in our set-up 1%), the probability that a workflow results correctly using WS-BA is decreased to roughly 60%, thus 40% lower than for AWM.

In sequential alignments $WP_{SEQ}$, the behavior of $p_{SA}(\omega)_{BA}$ with increasing numbers of elements greatly differs. In some experiments, $p_{SA}(\omega)_{BA}$ decreases with increasing number of elements. In other experiments the opposite is true: $p_{SA}(\omega)_{BA}$ decreases with ascending size of the workflow $n$. Thus, in sequential alignments, $n$ is not substantially decisive for $p_{SA}(\omega)_{BA}$. For all experiments, we are able to confirm the results using the analytical approach applying Formulas 8.7 and 8.8 to determine $p_{SA}(\omega)_{BA}$. The influential parameters for $p_{SA}(\omega)_{BA}$ is investigated in Sections 8.2.2.2 and 8.2.2.3.

**D.5 General Evaluation of the Order-to-Delivery Process**

**Test Set-Up** In this series of tests, we vary the transactional properties of the dynamically bound elements, thus the elements of the $WP_{XOR}$ patterns $XorVendor, XorDelivery$ and $XorPay$, $p_{RC}$ and $p_{RD}$. Additionally, we vary the success probability $p_S$ of these services. We assume three elements per $WP_{XOR}$ pattern to be present, thus $i = j = k = 3$. According to these settings, the workflow is filled with dynamically bound elements and the resulting degree of autonomy $d_{AWM}(\omega)$ and $d_{AT}(\omega)$ as well as the semi-atomicity probability $p_{SA}(\omega)_{AWM}$ and $p_{SA}(\omega)_{BA}$ are evaluated. For each depicted experiment, the results represent average values of at least 100 runs.

---

3This behavior is further discussed in the series of tests varying the success probability $p_S$, see Section 8.2.2.3
**Assumption**  Regarding the *degree of autonomy*, we assume the following: Based on our analytical model as well as the previous evaluation results, we expect similar trends of the degree of autonomy $d_{AWM} (\omega)$ as those of Section 8.3.1 (MoP). That is, we assume $d_{AT} (\omega)$ to be zero for all tests when using WS-AT. Employing AWM, if none of the dynamically bound services is recoverable and none is redoable (i.e., $p_{RC} = p_{RD} = 0$), all of them have to be coordinated, thus: $d_{AWM} (\omega) = 1 - 9/21 \approx 0.57$. With increasing ratios $p_{RC}$ and $p_{RD}$, $d_{AWM} (\omega)$ increases just as in Figure 8.9. As soon as either all dynamically bound elements are recoverable ($p_{RC} = 1$) or all of them are redoable ($p_{RD} = 1$), the resulting degree of autonomy $d_{AWM} (\omega)$ equals 1.

Considering the *semi-atomicity probability*, we expect similar trends for the resulting correctness guarantees $p_{SA} (\omega)_{AWM}$ and $p_{SA} (\omega)_{BA}$ as for MoP (see Figure 8.10). That is, $p_{SA} (\omega)_{AWM}$ is 1 for all settings. $p_{SA} (\omega)_{BA}$ increases with increased ratio of recoverable elements $p_{RC}$ to approach 1. Additionally, $p_{SA} (\omega)_{BA}$ exposes the characteristic behavior (decrease to a global minimum and then increase to 1) with increasing $p_{S}$.

**Evaluation**  In Figure D.7, we depict the resulting $d_{AWM} (\omega)$ (and $d_{AT} (\omega)$) on the y-axis for fixed values of $p_{RD} = 0.5, 0.8$ and $p_{RD} = p_{RC}$ and varied $p_{RC}$ on the x-axis. As assumed, the trends of the resulting degrees are similar to those for MoP (in Figure 8.9). If $p_{RD} = p_{RC}$, $d_{AWM} (\omega)$ increases from $d_{AWM} (\omega) \sim 0.57$ (for $p_{RC} = 0$) to 1 which meets our assumption.\(^4\)

![](image)

Figure D.7: $d_{AWM}$ of the order-to-delivery process, varying $p_{RC}$ and $p_{RD}$.

On the left hand side of Figure D.8, the ratio of recoverable elements among the dynamically bound services $p_{RC}$ is varied on the x-axis. On the y-axis the resulting semi-atomicity probability for different success probabilities of services ($p_{S} = 0.5, 0.9$) are depicted. The course of $p_{SA} (\omega)_{BA}$ is similar to the one in the MoP process (cf. Figure 8.10). If the success probability of the dynamically bound services is $p_{S} = 0.9$, the resulting $p_{SA} (\omega)_{BA}$ is roughly 1 for all values of $p_{RC}$. That relies on the fact, that with three branches in each $WP_{XOR}$ pattern, the chance that an $WP_{XOR}$ pattern completes

\(^4\)As the results for varied $p_{RD}$ and fixed $p_{RC}$ expose the same characteristics, we omit their presentation.
D.5 General Evaluation of the Order-to-Delivery Process

is: \(0.9 + (1 - 0.9) \times 0.9 + (1 - 0.9)^2 \times 0.9 = 0.999\). Thus, the chance that the whole workflow completes is extremely high.

If the success probability is \(p_S = 0.5\) and no dynamically bound element is recoverable or redoable, the chance that the order-to-delivery process semi-atomically completes is approximately 60%. If the ratio of recoverable elements \(p_{RC}\) increases, the resulting semi-atomicity probability is raised (to 1 for \(p_{RC} > 0.9\)). AWM ensures correctness independent of the success probabilities of the involved elements, thus \(p_{SA}(\omega)_{AWM} = 1\).

![Graph](image)

Figure D.8: \(p_{SA}\) of the order-to-delivery process, varying \(p_{RC}\) and \(p_S\).

On the right hand side, we depicted \(p_{SA}\) on the y-axis for \(p_{RC} = 0.5, 0.9\). \(p_{SA}(\omega)_{BA}\) proceeds characteristically, as detailedly evaluated in the empirical evaluation 8.2.2.3. For the order-to-delivery process, the global minimum of \(p_{SA}(\omega)_{BA}\) is reached for roughly \(p_{RC} = 0.3\). If half of the dynamically bound elements are recoverable, WS-BA ensures correct execution in only \(\sim 1/3\) of the all cases.

Conclusion  The results of these tests affirm the results achieved by the general evaluation (Section 8.2.1) and the evaluation of the MoP process (Section 8.3.1). Both \(p_{RC}\) and \(p_{RD}\) influence the number of conflicts and thus resulting \(d_{AWM}(\omega)\). For the order-to-delivery process, if only half of the included elements are redoable, AWM is able to increase the degree of autonomy \(d_{AWM}\) by approximately 90% as opposed to WS-AT.

Regarding the resulting correctness guarantees, the order-to-delivery example is as well illustrative for our previously achieved results. Using WS-BA, the correctness of the executed workflow is strongly dependent on the ratio of recoverable \(p_{RC}\) elements: The lower \(p_{RC}\), the lower \(p_{SA}(\omega)_{BA}\). If the success probability of the bound elements is \(p_S = 0.5\) and only half of the dynamically bound elements are recoverable (\(p_{RC} = 0.5\)), WS-BA ensures correct execution in only 65% of all cases.

When varying the success probability of services, the distinctive process of \(p_{SA}(\omega)_{BA}\) is observed: At first, the chance for semi-atomic failures mainly influences the resulting \(p_{SA}(\omega)_{BA}\). Thus, \(p_{SA}(\omega)_{BA}\) decreases to a global minimum of \(p_{SA}(\omega)_{AWM} \sim 0.3\) (\(p_S = 0.3\)). With further increase of \(p_S\), the chance for successful completion of the workflow is decisive for \(p_{SA}(\omega)_{BA}\): The semi-atomicity probability using WS-BA increases until it
approaches 1. AWM guarantees correct execution in all simulated settings.
Zusammenfassung


Analytische und experimentelle Ergebnisse bestätigen die Vorteile unseres Ansatzes: Im Gegensatz zu pessimistischen Verfahren, die zwar korrekte Ausführung garantieren, allerdings auf Kosten der Unabhängigkeit der Teilnehmer, erhöht unser Ansatz die Autonomie der Teilnehmer beträchtlich. Im Vergleich zu optimistischen Ansätzen, ermöglicht unser Ansatz, verschiedene (auch nicht kompensierbare) Dienste zu integrieren, ohne dabei die Korrektheit der Ausführung zu beeinträchtigen. Zusammenfassend lässt sich sagen, dass unser Ansatz ist ein hybrider Ansatz ist, der Korrektheit in jedem Fall und Autonomie so weit wie möglich garantiert.
F Erklärung


Berlin, den 28. April 2010

Katharina Hahn