BUSINESS PROCESS ARCHITECTURES

CONCEPTS, FORMALISM, AND ANALYSIS

RAMI-HABIB EID-SABBAGH

BUSINESS PROCESS TECHNOLOGY GROUP
HASSO PLATTNER INSTITUTE , UNIVERSITY OF POTSDAM
POTSDAM, GERMANY

DISSERTATION
ZUR ERLANGUNG DES GRADES EINES
DOKTORS DER NATURWISSENSCHAFTEN
– DR. RER. NAT. –

March 2015

ABSTRACT

Business Process Management has become an integral part of modern organizations in the private and public sector for improving their operations. In the course of Business Process Management efforts, companies and organizations assemble large process model repositories with many hundreds and thousands of business process models bearing a large amount of information. With the advent of large business process model collections, new challenges arise as structuring and managing a large amount of process models, their maintenance, and their quality assurance.

This is covered by business process architectures that have been introduced for organizing and structuring business process model collections. A variety of business process architecture approaches have been proposed that align business processes along aspects of interest, e. g., goals, functions, or objects. They provide a high level categorization of single processes ignoring their interdependencies, thus hiding valuable information. The production of goods or the delivery of services are often realized by a complex system of interdependent business processes. Hence, taking a holistic view at business processes interdependencies becomes a major necessity to organize, analyze, and assess the impact of their re-/design. Visualizing business processes interdependencies reveals hidden and implicit information from a process model collection.

In this thesis, we present a novel Business Process Architecture approach for representing and analyzing business process interdependencies on an abstract level. We propose a formal definition of our Business Process Architecture approach, design correctness criteria, and develop analysis techniques for assessing their quality. We describe a methodology for applying our Business Process Architecture approach top-down and bottom-up. This includes techniques for Business Process Architecture extraction from, and decomposition to process models while considering consistency issues between business process architecture and process model level. Using our extraction algorithm, we present a novel technique to identify and visualize data interdependencies in Business Process Data Architectures. Our Business Process Architecture approach provides business process experts, managers, and other users of a process model collection with an overview that allows reasoning about a large set of process models, understanding, and analyzing their interdependencies in a facilitated way. In this regard we evaluated our Business Process Architecture approach in an experiment and provide implementations of selected techniques.

ZUSAMMENFASSUNG

Geschäftsprozessmanagement nimmt heutzutage eine zentrale Rolle zur Verbesserung von Geschäftsabläufen in Organisationen des öffentlichen und privaten Sektors ein. Im Laufe von Geschäftsprozessmanagementprojekten entstehen große Prozessmodellsammlungen mit hunderten und tausenden Prozessmodellen, die vielfältige Informationen enthalten. Mit der Entstehung großer Prozessmodellsammlungen, entstehen neue Herausforderungen. Diese beinhalten die Strukturierung und Organisation vieler Prozessmodelle, ihre Pflege und Aktualisierung, sowie ihre Qualitätssicherung.

Mit diesen Herausforderungen befassen sich Geschäftsprozessarchitekturen. Viele der aktuellen Geschäftsprozessarchitekturen ordnen Geschäftsprozesse nach bestimmen Aspekten von Interesse, zum Beispiel, nach Zielen, Funktionen, oder Geschäftsobjekten. Diese Herangehensweisen bieten eine sehr abstrakte Kategorisierung von einzelnen Geschäftsprozessen, wobei sie wichtige Abhängigkeiten zwischen Prozessen ignorieren und so wertvolle Informationen verbergen. Die Produktion von Waren und das Anbieten von Dienstleistungen bilden ein komplexes System untereinander abhängiger Geschäftsprozesse. Diesbezüglich ist es unabdingbar eine ganzheitliche Sicht auf Geschäftsprozesse und ihre Abhängigkeiten zu schaffen, um die Geschäftsprozesse zu organisieren, zu analysieren und zu optimieren. Die Darstellung von Geschäftsprozessabhängigkeiten zeigt versteckte und implizite Informationen auf, die bisher in Geschäftsprozesssammlungen verborgen blieben.

In dieser Arbeit stellen wir eine Geschäftsprozessarchitekturmethodik vor, die es erlaubt Geschäftsprozessabhänigigkeiten auf einer abstrakten Ebene darzustellen und zu analysieren. Wir führen eine formale Definition unserer Geschäftsprozessarchitektur und entsprechende Korrektheitskriterien ein. Darauf aufbauend stellen wir Analysetechniken für unsere Geschäftsprozessarchitektur vor. In einem Anwendungsrahmenwerk eläutern wir die top-down und bottom-up Anwendung unserer Geschäftsprozessarchitekturmethodik. Dies beinhaltet die Beschreibung von Algorithmen zur Extraktion von Geschäftsprozessarchitekturen und zur Generierung von Prozessmodellen aus Geschäftsprozessarchitekturen, die die Konsistenz zwischen den Elementen auf Prozessmodellebene und Geschäftsprozessarchitekturebene gewährleisten. Aufbauend auf dem Extraktionsalgorithmus, stellen wir eine neue Technik zur Identifizierung, Extraktion, und Visualisierung von versteckten Datenabhängigkeiten zwischen Prozessmodellen in Geschäftsprozessdatenarchitekturen vor.

Unsere Arbeit stellt Geschäftsprozessexperten, Manager, und Nutzern einer Geschäftsprozessmodellsammlung eine Methodik zur Verfügung, die es ihnen ermöglicht und vereinfacht, eine Übersicht über Prozesse und ihren Abhängigkeiten zu erstellen, diese zu verstehen und zu analysieren. Diesbezüglich haben wir unsere Geschäftsprozessarchitekturmethodik in einem empirischen Experiment auf ihre Anwendbarkeit und Effektivität untersucht und zur weiteren Evaluierung ausgewählte Algorithmen implementiert.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

- Rami-Habib Eid-Sabbagh, Marcin Hewelt, Andreas Meyer, and Mathias Weske. Deriving Business Process Data Architectures from Process Model Collections. In *ICSOC 2013*. Springer Berlin Heidelberg, 2013.
- Rami-Habib Eid-Sabbagh and Mathias Weske. From Process Models to Business Process Architectures: Connecting the Layers. In *9th. International Workshop on Engineering Service-Oriented Applications (WESOA 13)*, pages 4–15, 2013.
- Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. A Tool for Business Process Architecture Analysis. In *ICSOC Demos 2013, Berlin, Germany, December 2-5, 2013*. Springer Berlin Heidelberg, 2013.
- Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. Business Process Architectures with Multiplicities: Transformation and Correctness. In *BPM*, volume 8094 of *LNCS*, pages 227–234. Springer, 2013.
- Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. Business Process Architectures with Multiplicities : Transformation and Correctness - Technical Report. Technical report, Hasso Plattner Institute, University of Potsdam, Potsdam, 2013.
- Rami-Habib Eid-Sabbagh and Mathias Weske. Analyzing Business Process Architectures. In *Advanced Information Systems Engineering*, volume 7908, pages 208–223. Springer Berlin Heidelberg, 2013.
- Rami-Habib Eid-Sabbagh, Matthias Kunze, Andreas Meyer, and Mathias Weske. A Platform for Research on Process Model Collections. In *BPMN2012 Workshop proceedings*, 2012.
- Rami-Habib Eid-Sabbagh, Remco M. Dijkman, and Mathias Weske. Business Process Architecture: Use and Correctness. In *BPM*, volume 7481 of *LNCS*, pages 65–81. Springer, 2012.
- Rami-Habib Eid-Sabbagh, Matthias Kunze, and Mathias Weske. An Open Process Model Library. In *Business Process Management Workshops (PMC2011)*, volume 100 of *LNBIP*, pages 26–38, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

The process of creation of the new depends essentially on the transgression of categorization.

—- Roald Hoffmann

ACKNOWLEDGMENTS

Writing a PhD is a journey with hills, flats, ups and downs, and in the end we reach the destination. Enjoy the journey, was a phrase that I often heard, and I did as I learned every day and I was accompanied by many wonderful and supportive people.

First of all, I am very grateful to my supervisor Mathias Weske for supporting me throughout my whole journey with guidance and encouragement, for his generosity, the freedom and all the opportunities that we enjoy at the Business Process Technology Group. He kept me on track and focused when I was about to take detours.

I am grateful to my reviewers Jan Mendling and Remco Dijkman. I have learned a lot from the research collaboration with Remco Dijkman who watered my young business process architecture research plant. I also enjoyed the research collaboration with Henrik Leopold and Jan Mendling in which I learned a lot about natural language processing.

My journey would have not been as wonderful without my fellow travelers and colleagues from the BPT group. My deepest thanks to the whole BPT group who created a great research atmosphere with challenging discussions, super companionship and mutual support. Thank you Ahmed, Alex, Andy, Andreas, Anne, Kimon, Katya, Luise, Matthias K., Matthias W., Marcin, Oleh, Thomas and Tiku. I loved the fruitful, intensive, and constructive discussions and exchange of ideas with you as well as the fun tabletop football matches and motoric skill training at lunch and think breaks. I very much enjoyed the research collaboration with my colleagues Marcin Hewelt, Matthias Kunze, Andreas Meyer, and Kimon Batoulis. Thank you Marcin, Matthias, Thomas, Andy, and Kimon for proofreading my thesis. As music has been my second motor every day of my work, I would like to thank all musicians that I listened to at work.

Finally, I am very thankful to my love Johanna and Mr. Karl, my parents Anneliese and Alfred, my sister Yasmine and my brother Karim, who all supported me throughout the whole time of my PhD research with inspiration and encouragement. Thank you Johanna for accompanying me and caring for me during this long travel with love, patience, and humor.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

BPA	Our Business Process Architecture
BPM	Business Process Management
BPMA	Business Process Model Abstraction
BPMN	Business Process Model and Notation
DBPA	Business Process Data Architecture
EPC	Event Driven Process Chain
OLC	Object Life Cycle

ON	Open net
PA	General Business Process Architecture
PMC	Process Model Collection
PN	Petri net
RPST	Refined Process Structure Tree
WF net	Workflow net

Part I

INTRODUCTION & BACKGROUND

1 INTRODUCTION

This chapter introduces our general motivation for developing a novel business process architecture approach and situates business process architecture research in the current Business Process Management research stream. It present motivational settings from the public and private sector to illustrate our research with examples. This chapter lists the problems tackled by our work, our research objectives, and the scientific contribution of this thesis. It concludes with providing an outline of the thesis.

The innovation in the IT-sector drives companies to adapt, act, and react to the changing needs and requirements of customers and new emerging markets. For a company to be able to adapt and adjust to the changing environment, a clear overview over its operations and their interdependencies is necessary.

Business Process Management (BPM) is a means to portray and improve a company's operations. Each production of a good or delivery of service can be reflected in one or several process models. BPM has gained attention in the private and public sector for improving an organization's operations by increasing the efficiency and reducing costs of its processes [114, 165].

In the course of BPM efforts, companies and organizations build up large process model repositories with many hundreds and thousands of processes bearing large amounts of information. With the advent of large business process model collections, new challenges arise as the need for structuring and managing the business process models, their maintenance, and their quality assurance, among others [34, 62].

In such large collections, it is difficult to get an overview of the processes of a company and how those processes relate to each other. Many large BPM projects fail as they miss coordination, quality assurance measures, or an encompassing methodology for the modeling of business processes. This leads to process model collections of poor quality [119, 101]. Business process architectures (PA) support the management of process models within a process model repository by defining the relations between business processes and providing guidelines for organizing them [35]. Many PA approaches classify the business processes within a process model repository along functions, business objects, or other aspects considering one process model at a time. Although providing a structure and overview on the process models within the process model repository such classification approaches keep valuable information of process models hidden. In many cases the current approaches lack a clear underlying model and formalism for structuring and analyzing business processes and their interrelations.

Hence, novel PA approaches for organizing and structuring business process models in process model repositories are required to complement existing hierarchical PA approaches, visualizing the hidden process information and increase the return on investment of BPM projects.

In this thesis we propose a novel PA approach that considers horizontal interdependencies between process models providing a holistic view on process model collections. Considering all process interdependencies leads to an end-to-end assessment of a larger scenario, e. g., a business foundation with all its preparatory processes for acquiring the necessary regulatory documents and its succeeding processes like tax payment registration.

This is the foundation for visualizing, connecting, and analyzing additional process models and their meta data under consideration of their interdependencies. The visualization of business process interdependencies on an abstract level shall provide a better overview for coordinating business processes and business process modeling projects. Hence, it shall improve the quality of the process model collection as the required output and input of processes is better captured and harmonized.

This chapter gives a brief introduction into Business Process Management and the role of PAs within Business Process Management in Section 1.1 and in Section 1.2. At the same time, we present PA use cases from the public and private domain that we will use to motivate and exemplify our research results. Section 1.3 presents the problems in PA research that we address with our work. Section 1.4 introduces our research objective and Section 1.5 summarizes our scientific contribution to research on PAs. Section 1.6 outlines the structure of this thesis.

1.1 BUSINESS PROCESS MANAGEMENT

Business process management has many facets and consists of methods and techniques for the design, management, implementation, execution, and analysis of business processes [165]. At its core is the improvement of a company's operations by explicitly representing activities and their control flow in business processes, and aligning and structuring the organization along those business processes [165, 36].

A business process consists of a set of coordinated activities that contribute to a specific business goal and is enacted by single organizations [165]. A business process can also interact with external business processes from other organizations in a business process choreography or with other internal business processes.

BPM projects may start from pressing needs such as budget cuts in the public administration, demographic changes (older employees retire), i. e., the need for increased efficiency with less employees, or as a general strategy set from the top management to stay atop of current developments, foster continuous improvement, and stay flexible to adapt to ever changing market requirements.

The development of business processes and their implementation in a software system is described by the BPM lifecycle [165]. The BPM lifecycle consist of four phases; Design and Analysis, Configuration, Enactment, and Evaluation. Figure 1 shows the BPM lifecycle and the involved roles during each phase.

The BPM lifecycle is entered in the *Design and Analysis* phase. During this phase business processes are identified, modeled, analyzed, validated and simulated. Usually (project) managers, process owners, business process experts, and domain experts are involved in this phase depending on the size of the BPM project. Ideally, modeling guidelines, a business process modeling notation, and a method to elicit and organize the business process models are also defined in this phase. The result of this phase are validated process models.

The second phase of the BPM lifecycle, the *Configuration* phase, encompasses the implementation of the business process in an IT-system by procedures and regulations or a combination thereof [165]. For the implementation of a business process model in an IT-system, the according platform and technology is chosen. During its implementation, the business process model is enriched with IT-specific aspects and technical details that they can be run on a process engine or chosen platform [165].

In the *Enactment* phase process instances are executed either by the help of an IT-system or just by the person assigned. This phase consists of the monitoring of the running process instances either by the monitoring component provided by the IT-system or other monitoring techniques in case of manual execution of process instances. In the monitoring component the state of the process instances and other information are visualized for tracking the status of each instance [165].

The *Evaluation* phase includes the evaluation of the process performance, analysis of compliance issues and other evaluation topics based on business process execution logs. For the evaluation, often advanced process mining techniques are used [165].

The main application area of PAs is the Re-/Design and Analysis phase which is highlighted in Figure 1. We extended the Design and Analysis phase of the BPM lifecycle from Weske [165] with required characteristics that a PA approach should provide in this phase. Rosemann [124] names lack of coordination and organization in large modeling projects as one major pitfall. Agreeing with this fact, Recker [120], Dijkman et al.[34, 31, 35], and Houy et al. [62] highlight the need for novel techniques for organizational modeling, managing

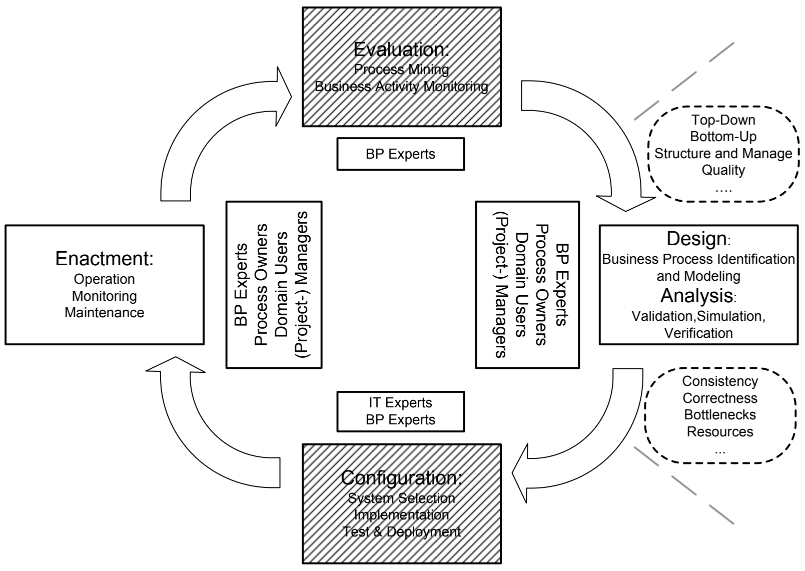


Figure 1: BPM lifecycle

and structuring large process model collections, and new methods for process model decomposition and abstraction that support this project phase.

PAs support the identification, the structuring and managing of process models, and the coordination of the business process elicitation phase. Depending on the PA approach, PAs may also support the analysis of process models. To some extent PAs support the enactment phase, e. g., the monitoring and maintenance of process models within a process repository. The structure of the business process model collection can be used for visualizing and aggregating process instance information. In Figure 1 we highlighted the Analysis and Design phase as the primary application area of PAs and to a lesser extent the enactment phase.

1.2 BUSINESS PROCESS ARCHITECTURES

Introducing BPM in a company and starting modeling business processes requires a set of methodological and preparatory actions, the definition of a modeling goal, specification of modeling guidelines, and system for classifying the process models, which are stored in a process model repository.

Business process architectures define the structure of a process model repository and interrelation between business process models for organizing the business process models within the process model collection [35]. They are a graphical visualization of the business processes of an organization and their relations with each other [31].

The terms business process landscapes, business process maps, value chains are used synonymously for business process architecture. In

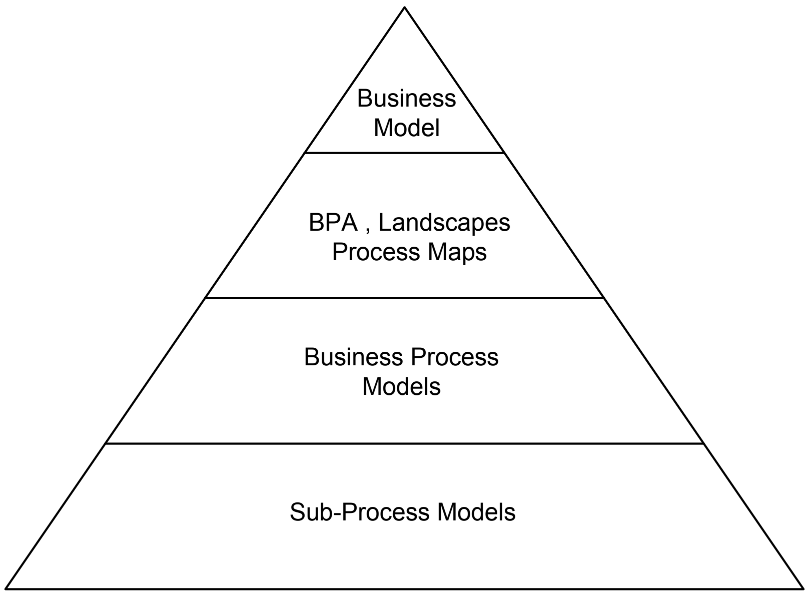


Figure 2: Business process repository layers

some cases a PA consists of several such views, e. g., value chain and process maps on different process model repository layers. Figure 2 shows a general structure of a business process model repository. The top layer defines the business model, i. e., the main services or products that generate value and are sought after by the customers. This layer is very abstract and provides a business perspective on the company. The business layer does not always exist in a process model repository. In such cases the PA layer is the top layer of the process model repository.

The PA visualizes the process model collection along aspects defined by the PA approach in use, e. g., business functions that are decomposed down to the process model layers. In this regard a PA approach defines the structure of the process model collection by structuring this overview layer and defining the connection to the lower layers of the process model repository and their elements. Many business process repository platforms provide value chains, process landscapes, or process maps on PA layer for structuring the process model collection.

The lower layers of a business process repository focus on process models and provide detailed descriptions of the business processes and their sub-processes. Figure 2 shows only a general structure of a business process repository whereas in reality the number of repository layers varies according to complexity and desired granularity.

1.2.1 *Motivational Use Cases*

To exemplify our business process architecture approach we consider use cases from the public and from the private sector.

PA use case in public sector The public sector consists of legally autonomous organizations with shared goals and a common budget. A public organization is divided into different departments that deliver public services to citizens, businesses, or other public organizations. Each department is responsible for a particular topic, e. g., finance, construction, or social welfare. In most cases, each department considers its public services as one business process and represents it in their common process model repository as one business process model[1] [42].

Examining the preconditions of the public services, we notice that many public services depend on each other as the output of one public service is the input to another at the beginning or during carrying out public services. The current PA approaches do not reflect such interdependencies. Process maps, landscapes and similar PA approaches do not provide sufficient information in this field. The different departments in the public sector know that they cooperate with other departments in regard to their public services, but hardly know when, why, and how. Taking a holistic view at the interdependencies of business processes is a major necessity to assure correct business process collaboration [56, 91, 10].

In this thesis, we will look at different such examples where a customer's need is served by the delivery of several processes that depend on each other. This can be an entrepreneur's desire to open a business for example. The founding of a new enterprise encompasses different public services, of which a selection is depicted as Event-Driven Process Chains models in Figure 3. Each public service for itself results in a desired outcome. It is not obvious that they depend on each other as it has been highlighted by the dashed boxes in Figure 3. From this perspective, we cannot say anything about how they relate to each other or about the correctness of their interaction, e. g., the successful founding of a new enterprise. To apply for a trade agent permit, an entrepreneur has supply a freedom of movement certificate. At the same time, the entrepreneur is required to provide evidence of a job employment when applying for a freedom of movement certificate. However, the document certifying his self employment can only be issued after the entrepreneur was registered as trade agent, so that he cannot apply for the freedom of movement certificate in the first place. The requirements and regulations of the public services contradict, although all by themselves are correct.

This example from the public sector shows the importance of considering, visualizing, and analyzing business process interdependencies and serves as part of our motivation. An overview of the interdependent processes has advantages for different user groups. Process owners can assess the impact of change for other involved proces-

1 EU Services Directive Realization in Berlin – https://www.ea.berlin.de/web/guest/home

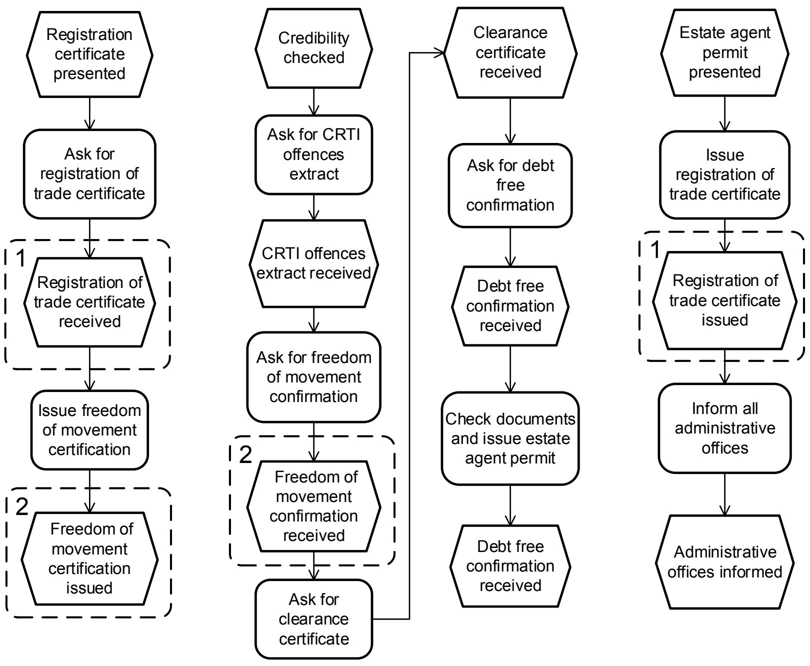

Registration certificate presented
Ask for registration of trade certificate
1
Registration of trade certificate received
Issue freedom of movement certification
2
Freedom of movement certification issued
Credibility checked
Ask for CRTI offences extract
CRTI offences extract received
Ask for freedom of movement confirmation
2
Freedom of movement confirmation received
Ask for clearance certificate
Clearance certificate received
Ask for debt free confirmation
Debt free confirmation received
Check documents and issue estate agent permit
Debt free confirmation received
Estate agent permit presented
Issue registration of trade certificate
1
Registration of trade certificate issued
Inform all administrative offices
Administrative offices informed

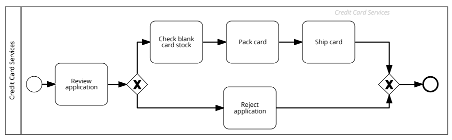
Figure 3: EPC business process models depicting three public services

ses, how their processes are influenced by other departments, and manage risks according to identified interdependencies for example. Employees can grasp the overall context of their work and identify interfaces to other processes and departments. Customers, if provided with such an overview, get the full picture and realize that their project requires the application to several public services and may save time by preparing and submitting all necessary input at once.

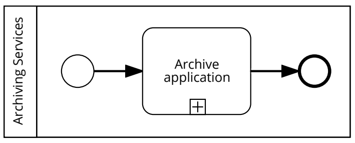
PA USE CASE IN PRIVATE SECTOR Our second use case derives from an online financial service provider from the private sector. Their main advantage in the market is a very efficient service delivery in contrast to financial service providers with face-to-face customer contact who rather build on the personal relation between service provider and customers.

The delivery of financial services like a credit card application internally encompasses many business processes. In a simplified way, the credit card application includes four business processes: customer identification, credit card application handling, invoicing process, and archiving; all of which are run by different departments. The exemplary business processes are depicted in Figure 4.

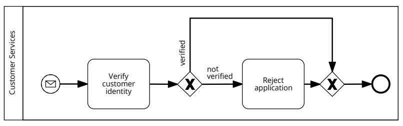
First the identity of the customer who applies for the credit cards needs to be verified, which is done by the "Customer identity verification" process in Figure 4c. If the provided documents are correct, the customer is verified, else the application gets rejected. If the customer is verified, the credit card application can be handled by the "Credit



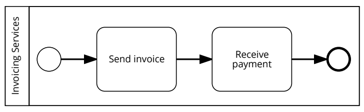
(a) Credit card application handling



(b) Archiving



(c) Customer identity verification



(d) Invoice handling

Figure 4: Process models of a credit card application

card application handling" process, depicted in Figure 4a. The credit card is either issued or the application is rejected due to a low customer score for example. At a certain point of time the credit card fees have to be paid which is handled by the invoice handling process in Figure 4d. At last, the application should be archived for which an archiving process exists, shown in Figure 4b. For the external customer the credit card application may appear as one process. However, from these process models it is not obvious how these business process models are related to each other. The identification of their interdependencies allows understanding the complete picture. Extending this scenario with data objects, we will describe a technique to identify the interdependencies between these processes in Chapter 8. Only the close adjustment of all internal processes leads to efficient service provisioning.

In the private sector, risk management is a major use case for the assessment of end-to-end processes. Companies that provide financial services are obliged to fulfill strict federal regulations and explicitly document their processes. These regulations are set by the federal financial supervisory authority (BaFin). Each business process must be verified regularly. For a consistent risk management, all business processes need to be verified that (in-)directly influence the process under consideration. Such a tasks can only be performed by looking at the big picture and analyzing all involved processes and their relations to each other. Risk management is one of many use cases in the financial sector where a holistic view on the business processes, their interdependencies, and their impact and influence is of utmost importance.

Our motivation, exemplified by the use cases from the public and private sector, is to improve the current state of research and to provide a PA approach that supports managers, chief information officers, auditors, and other stakeholders in understanding the complex system of their interdependent processes and support their operative and strategic planning and analysis. According to the Forbes article from 2012 “The Top 10 Strategic CIO Issues For 2013”[2], the comprehension of end-to-end process aspects is one of the strategic issues that will provide a significant advantage for companies and organizations [83].

1.3 PROBLEM STATEMENT

Companies develop collections of hundreds or thousands of business process models that represent the complex system of cooperating entities that form an organization. Designing and analyzing the structure of this system of business processes emerges as a new challenge, which is addressed by the field of business process architectures.

The research on PA, i. e., on the structuring and organization of process models within a process model collection, has produced initial ideas, especially in the area of hierarchical classification of process models. A range of valuable approaches have been proposed so far that are commonly used in process model repositories. However, we identified the following issues that should be overcome for improvement:

- Loss of process orientation on PA level in most approaches
- Lack of holistic end-to-end process view
- Lack of formal framework and desired properties for PA
- Lack of analysis techniques
- Hiding of valuable process information

Common PA approaches classify process models along different aspects of interest on a very high level of abstraction. By classifying processes, the process orientation on PA level is lost, and important information for the design, optimization, and analysis of business processes stays hidden. Much attention has been paid to approaches focusing on single processes, but the larger context of process models in regard to their interdependencies has been neglected so far.

The larger context depicted by all interdependent process models is important for reasoning on the impact of changes, identifying key processes, and determining resource requirements as well as evaluating process cost and other aspects of interest.

An approach focusing on multiple business processes to study the complex system of interdependent processes within an organization and their influence on each other does not exist. Existing PA approaches miss measures and properties for the assessment of the process

2 Accessed: 25th April 2014 - http://www.forbes.com/sites/oracle/2012/09/28/the-top-10-strategic-cio-issues-for-2013/

interdependencies and the quality of coordination of the business process models within a process model collection.

While simple forms of relationships have been presented, real-world scenarios show that complex relationships between business processes are rather the rule than the exception. So far, the actual PA approaches do not provide a framework to capture, visualize and analyze those relationships.

The treasure of information depicted in process models is hidden on higher level by common PA approaches. Considering process interdependencies, this information can be combined and aggregated creating additional value.

1.4 RESEARCH OBJECTIVE

Our research explores the complex system of business processes that represent an organization. In the relevant PA research like for example [35, 70, 54], we noted that there are only few PA approaches that establish a holistic view and provide larger context on business processes and their interdependencies. Even less PA approaches provide a formal underpinning that allows for the definition of quality properties and analysis of PAs.

We present a novel PA approach that has a formal foundation, allows for defining and specifying the interrelations between business processes within a process model collection, visualizing and analyzing them. On top of the foundational concepts, we define initial quality properties for our BPA approach that allow for a better adjustment of interdependent processes.

Our research consist of the following steps:

1. The collection and definition of requirements
2. The formal definition of the basic concepts of our BPA approach
3. The definition of desired properties and analysis concepts
4. The application and evaluation of our BPA methodology

As we expect our BPA approach also to be used by non-BPM-experts as managers and project leaders, among others, the main concepts shall be simple, easily comprehensible, and facilitate the understanding and visualization of a complex system of interdependent business processes and their influence on each other.

In the following, we refer to our approach as *Business Process Architecture* (capitalized) and abbreviated as *BPA* whereas we refer to the general field of business process architectures as *business process architecture* (lower case) and abbreviated as *PA* for better reading and differentiation of those two terms.

1.5 SCIENTIFIC CONTRIBUTION

This work contributes a novel and simple approach to the PA research field. The approach is complementary to most existing approaches and provides a novel holistic overview on business processes. Considering horizontal business process interdependencies is especially important for coordinating required and expected process inputs/outputs, and message exchanges, and examining the impact of restructuring and optimizing business processes.

The scientific contribution of this thesis consists of the following parts.

- The formal description of the elements and the relationships forms the heart of our BPA approach. The formal concept allows for the development of BPA correctness criteria and their analysis (see Chapter 4 and Chapter 5).

- The definition of structural and behavioral correctness criteria for BPAs. So far none of the existing PA approaches describe desired and undesired properties when organizing business process models within a business process repository (see Chapter 4).

- Unique Business Process Architecture analysis techniques that allow analyzing created BPAs for their correctness and support the quality improvement of business process model collections (see Chapter 6). Because of lacking PA properties in current approaches no analysis techniques exist. Few surveys exist that provide questions for evaluating PA approaches but not PAs themselves.

- An overall methodology for the application of our BPA concept. The methodology describes the use of our BPA approach bottom-up and top-down (see Chapter 7). This includes techniques for the bottom-up BPA extraction from a process model collection (see Section 7.2) and top-down decomposition from BPA to process models (see Section 7.3).

- An extraction technique considering data objects that allows for extracting data BPA based on data interdependencies found in business process models (see Chapter 8).

Our work adds to the existing PA research and fills some gaps in the area of multi-process focused PAs with formal foundations and analysis techniques of which none exist yet. The work is a starting point for providing more expressive and informational PAs and according overviews. It lays the formal foundations for a novel BPA concept and a set of analysis and application techniques.

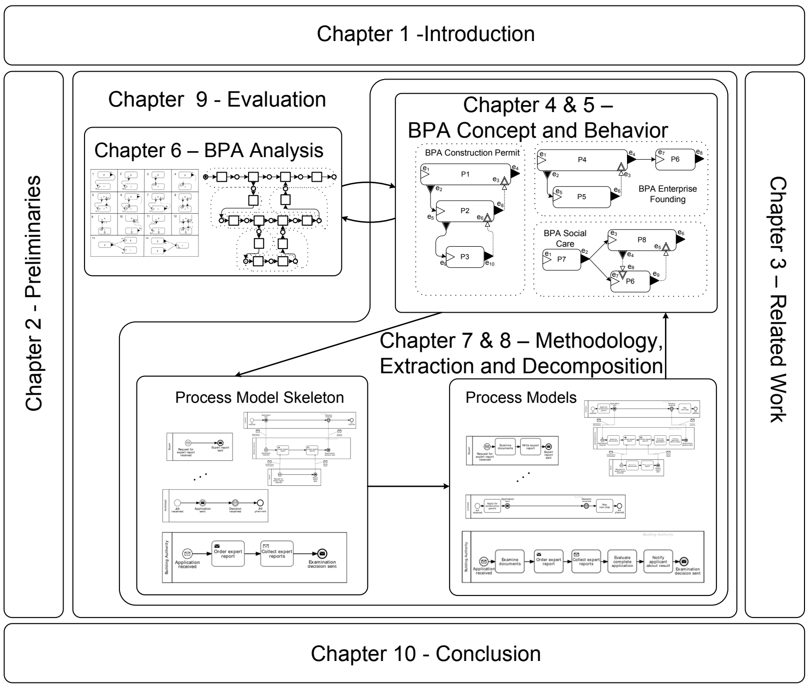


Figure 5: Thesis structure

1.6 OUTLINE OF THESIS

This work consists of four major parts, introduction and background, conceptual design, extended concepts and application; and evaluation and conclusion. The structure and the overall picture of our thesis is depicted in Figure 5.

The first part captures preliminaries (Chapter 2) and related work (Chapter 3) and introduces existing concepts and techniques that we use or refer to in this work. In the related work chapter, we present a broad range of approaches that relate to the field of PAs and the PA approach that we present in the following chapters. We discuss current trends and identify gaps in PA research, as well as requirements on which we build our BPA approach. It serves as background on existing approaches and for situating our work in the PA and BPM research field. These chapters and the conclusion chapter in Chapter 10 form the frame of our work.

The second part presents the conceptual design of our BPA approach, its elements, the semantics of the relationships depicted (Chapter 4), and defines their behavior and properties (Chapter 5) and analysis techniques (Chapter 6). It describes the core techniques of our BPA approach.

The third part provides insight into extended concepts and application of our BPA approach. We describe the overall bottom-up and top-down application of our BPA approach in a methodology, introduce the necessary BPA extraction and BPA to process model decomposition techniques (Chapter 7), and provide techniques to identify and extract data interdependencies between business processes (Chapter 8).

In the last part, we present a two-folded evaluation of our BPA approach (Chapter 9). First, we describe an experiment in which we evaluated our approach with practitioners for its applicability, ease of use, and usefulness. Second, we provide insights on prototypical implementations of our BPA approach, BPA correctness analysis, and the extraction of BPAs from existing process models. Subsequently, we discuss the contributions of our BPA approach, their limitations and open issues, and provide an outlook on future research (Chapter 10).

2 PRELIMINARIES

This chapter provides the foundations for our work. We present Business Process Management and workflow concepts that are used and referred to in later parts of this work.

This chapter introduces BPM concepts, notations, and formalisms as foundation and for the description of the context of our business process architecture approach. We first introduce business process orchestrations and business process choreographies as PAs relate to those concepts. In this regard, we focus on the Business Process Model and Notation (BPMN) with its different concepts and diagram types and Event-Driven Process Chains (EPC) as they are quasi-standards in the industry for modeling business processes. We introduce Petri nets and some of its sub-classes, i. e., Workflow nets, Workflow modules, and Open nets and elaborate on formal analysis concepts for workflow notations.

2.1 BUSINESS PROCESS ORCHESTRATIONS

A business process orchestration defines the internal/private behavior of a business process. In literature, business process orchestrations are also referred to as business process, workflow, BPM process, or orchestration of services [112]. A business process orchestration describes the internal control flow structure and provides details on the execution constraints of activities and the occurrence of events of a process [165]. Resource and data information can also be depicted in business process orchestrations. Common control flow structures of processes have been described in an extensive list of control flow patterns in Aalst et al. [153] that can be used to assess the expressiveness of a process modeling notation [153].

Many formal and graphical process modeling notations have been proposed. The most prominent are Business Process Model and Notation (BPMN 2.0), Event-Driven Process Chains (EPC), workflow nets, Yet Another Workflow Language (YAWL), Business Process Execution Language (BPEL), and Petri nets that are commonly used to provide formal semantics for some of the aforementioned business process modeling notations [80].

2.1.1 *Process Model*

Business process orchestrations are depicted in process models. To be able to refer to a process model in a general but formal way we introduce the definition of a process model in the following. It consists of nodes that are events, activities, gateways; data objects, and sequence flows and data flows [165].

Definition 1 (Process Model) *A* process model PM *is a tuple* $(N, \mathcal{D}, CF, DF, type)$ *in which:*

- $N = A \cup G \cup E$ *is the set of nodes being activities* A*, gateways* G*, and events* E *where* A, G, E *are pairwise disjoint*
- $CF \subseteq N \times N$ *is the sequence flow relation, such that* (N, CF) *is a connected graph*
- $\mathcal{D}$ *is a finite, non-empty set of data objects, with* $N \cap \mathcal{D} = \emptyset$
- $DF \subseteq (\mathcal{D} \times (A \cup E)) \cup ((A \cup E) \times \mathcal{D})$ *describes the data flow relation, by depicting a node's reading from respectively writing to data objects*
- $\bullet n, n\bullet$ *describe the preset, respectively postset, of a node* n *that contains the predecessor, respectively successor nodes of* n*, such that* $\bullet n = \{x | (x, n) \in CF\}$ *and* $n\bullet = \{y | (n, y) \in CF\}$
- *type* : $G \rightarrow \{xor, and\}$ *assigns a type to each gateway.* ◇

2.1.2 *Data Aspects in Process Models*

Data in process models is usually represented by data objects and their states. A data object is an entity processed during process execution that is characterized by its states and state transitions. In the following we define data objects and data states as follows.

Definition 2 (Data Object and Data State) *A* data object $\mathcal{D}$ *is an entity or any piece of information or physical item being processed, manipulated, or worked with during business process execution. Each data object is in exactly one data state at one point in time.* Data states *represent the results of processing a data object in the process context. Thereby, each data state describes a specific situation of interest to the organization from the data object's point of view.* ◇

The states of data objects and the transitions from one state of the data object to the other can be expressed by *object life cycles (OLC)*. In this work, we consider only *acyclic* OLCs. Thus, all paths, i. e., sequences of states and transitions from the initial to the final states are of finite length. Different paths reflect different possibilities to operate on the data object [43]. An exemplary OLC is depicted in Figure 6.

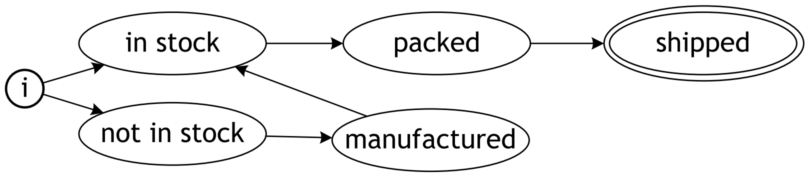


Figure 6: Exemplary data object life cycle

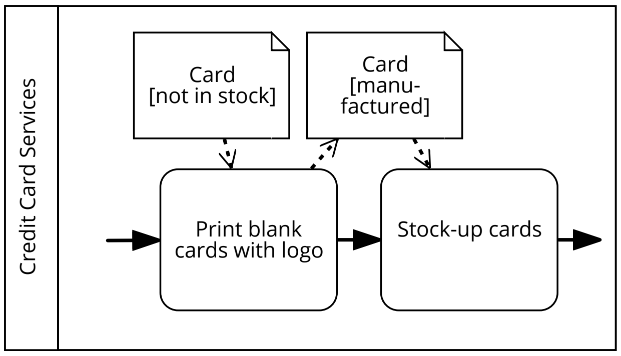


Figure 7: Use of data objects in a process model

Definition 3 (Object Life Cycle) *An object life cycle is a tuple* $OLC = (S, i, S_F, T_D, \Sigma, \eta)$ *that consists of:*

- S *a finite set of data states*
- $i \in S$, *an initial data state*
- $S_F \subseteq S$ *a non-empty set of final data states*
- $T_D \subseteq S \times S$ *a finite, acyclic relation of data state transitions*
- Σ *a finite set of actions representing the manipulations allowed on the corresponding data object*
- $\eta : T_D \to \Sigma$ *is a function that assigns an action to each data state transition* ◇

We consider data objects act as pre- and postconditions of activities. An activity is enabled and can subsequently be executed, only if all its input data objects (pre-condition) are in the specified state. At proper termination, the activity sets its output data objects into the specified state, the postcondition of the activity, as depicted in Figure 7. However, when one activity has the same data object $\mathcal{D}$ several times as precondition, each time in a different state, then $\mathcal{D}$ can be in any of these states to enable the activity. Similarly, if an activity has the same data object in different states as output, the activity sets the data object in one of the specified state at proper termination.

When dealing with data in process models, we assume that each data state transition is realized by at least one activity from a process model and that all data accesses of activities are modeled and that no activity writes a data object it has not read before. Data objects in the postcondition must also occur in the precondition.

The above introduced definitions will be used for the extraction of data BPAs in Chapter 8.

2.1.3 *Decomposition of Process Models*

A large body of research deals with the decomposition of process models into fragments for analysis, abstraction or transformation. Different techniques have been proposed, e. g., process structure trees or refined process structure trees [159, 160]. In this thesis we will rely on *refined process structure tree* (RPST), a technique for hierarchically decomposing process models into fine grained *single entry/ single exit* (SESE) fragments. The RPST technique has several advantages, first the resulting fragments can be structurally isolated as sub processes and second two fragments either contain each other or do not intersect [136] . An RPST provides an efficient way to parse process models and produce a hierarchical tree of SESE fragments. A SESE fragment is defined by two boundary nodes, that if removed, disconnect the fragment from the process model [115, 136]. All fragments on one level of the RPST are disjoint. A SESE fragment, however, contains all its child fragments on the lower level. Hence, the root node of the RPST contains all SESE fragments of the process model. These fragments are fine grained, modular, and canonical [160, 136].

To define the RPST we first need to introduce the concept of fragments, boundary nodes, components and canonical components. We adapt the definition of Smirnov [136] to also consider events in a process model. As this technique can only be applied to block structured process models with a single start and a single end node we introduce a normalized process model definition by defining a single start and a single end node as well as restrictions to gateway nodes. In Chapter 8 we show how to include data associations for the abstraction of process models in regard to data interdependencies by mapping them to according control flow structures and hence implicitly consider in the decomposition of process models.

Definition 4 (Normalized Process Model) *Let* $nPM = (N, CF, type, s, f)$ *be a process model with a single start node* s *and a single end node* e*, where* $s, f \in A \cup E$.

- $s \in N \setminus G$ *with* $\bullet s = \emptyset$ *denotes the only start node of a process model, i. e.*$\forall n \in N : \bullet n = \emptyset \Leftrightarrow n = s$
- $f \in N \setminus G$ *with* $f\bullet = \emptyset$ *denotes the only end node of a process model, i. e.*$\forall n \in N : n\bullet = \emptyset \Leftrightarrow n = e$
- $\forall n \in N \setminus G : |\bullet n| \leq 1 \wedge |n\bullet| \leq 1$

- $\forall g \in G : (|\bullet g| = 1 \wedge |g\bullet| > 1) \vee (|\bullet g| > 1 \wedge |g\bullet| = 1)$ ◇

The process model notion for decomposing process models does not allow events and activities with postsets or presets larger one, i. e., multiple incoming or outgoing edges. Only gateways are allowed to have either one predecessor node and several successor nodes or vice versa, i. e., either one incoming and several outgoing edges or several incoming and one outgoing edge respectively. To be able to use the research on business process model abstraction of Smirnov [136, 135] that maps process models to graphs we introduce graphs in the following.

Definition 5 (Directed Graph) *A graph is a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with $\mathcal{N}$ being a non-empty set of nodes and $\mathcal{E} = \mathcal{N} \times \mathcal{N}$ being a is a set of edges, connecting ordered pairs of elements of $\mathcal{N}$.* ◇

The set of incoming edges $in(n)$ of a node $n \in \mathcal{N}$ of the directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is defined as $in(n) = \{(n', n) | n' \in \bullet n\}$ and the set of outgoing edges of node n as $out(n) = \{(n, n') | n' \in n\bullet\}$ [136].

Definition 6 (Process Model Fragment [136]) *Let $nPM = (N, CF, type, s, f)$ be a normalized process model. The tuple $PMF = (N_f, CF_f, type_f)$ denotes a fragment of process model nPM, where (N_f, CF_f) is a connected subgraph of the graph (N, CF) and the function $type_f$ is the restriction of type of PM to set G_f.* ◇

We define a boundary node according to [136] as follows.

Definition 7 (Boundary Node [136]) *Let $nPM = (N, CF, type, s, f)$ with $PMF = (N_f, CF_f, type_f)$ be a process model with a process model fragment.*

- *A node $n \in N_f$ is a* boundary node *of PMF if $\exists f \in in(n) \cup out(n) : f \notin CF_f$*
- *A node n is an* entry *of PMF, if it is a* boundary node *and $in(n) \cap CF_f = \emptyset$*
- *A node n is an* exit *of PMF, if it is a* boundary node *and $out(n) \cap CF_f = \emptyset$* ◇

Definition 8 (Component [136]) *Let $nPM = (N, CF, type, s, f)$ be a process model with a process model fragment $PMF = (N_f, CF_f, type_f)$. The fragment PMF is a* component *if it has exactly two boundary nodes; one entry and one exit node. By $\mathbb{F}$ we denote the set of all components in a nPM [136].* ◇

Definition 9 (Canonical Component [136]) *We call a component $PMF = (N_f, CF_f, type_f)$* canonical *if $\forall PMF' \in \mathbb{F} : PMF \neq PMF' \Rightarrow ((CF_f \cap CF_{f'} = \emptyset) \vee (CF_f \subset CF_{f'}) \vee (CF_{f'} \subset CF_f))$. Δ denotes the set of all canonical components of a process model nPM.* ◇

According to [160, 136], each of the canonical components can be classified into four types, trivial, polygon, bond, and rigid. A single edge depicts a component of trivial type TR. A polygon PO depicts a sequence of nodes or a sequence of components. A bond BO consists of components that share common boundary nodes. A component that does not classify for one of the above is called a rigid RI. The function component-type ct assigns a type to each component, such that $\mathsf{ct} : PMF \to \{TR, PO, BO, RI\}$ Bonds that are bound by AND or XOR gateways are referred to as XOR respectively AND bonds [160, 136]. The RPST is defined as follows.

Definition 10 (Refined Process Structure Tree (RPST) [136]) *Let* $nPM = (N, CF, type, s, f)$ *be a process model. An arborescence* $RPST_{PM} = (\Delta, r, \xi)$ *is the refined process structure tree of a process model* nPM*, such that*

- Δ *is the set of all canonical components of* nPM
- r *denotes a component that is the root of a tree*
- $\xi \subseteq \Delta \times \Delta$ *depicts the parent child relation between a component and one of its children* ◇

The construction of an RPST for a process model takes time linear in regard to the number of nodes in the process model [136]. In Chapter 7 the technique of RPST will be used to describe characteristics of process models that can be extracted to BPAs, to describe the BPA extraction algorithm, as well as the generation of process model skeletons from BPAs.

2.2 BUSINESS PROCESS CHOREOGRAPHIES

A business process choreography defines the inter-organizational communication interaction between two or more business processes. It can be viewed as business contract that specifies the public behavior that two or more collaborating entities agreed to. The internal or private processes are hidden as companies regard them as a valuable asset that they keep secret.

Process choreographies can be modeled with different process model diagram types. BPMN offers business process diagrams and choreography diagrams to model such an interaction. Choreography diagrams abstract completely from the internal behavior of processes and only depict the message interaction behavior between the processes that are represented as participants.

Business process diagrams allow modeling the internal business process, as well as the message flows between two collaborating processes. Section 2.3 will elaborate further on BPMN2.0 and its different diagram types. Beside BPMN, rather formal notations such as Open nets, interaction Petri nets, and Workflow modules, allow the modeling of message exchange between processes.

When two business processes collaborate their structure and behavior needs to be compatible to each other to ensure correct interplay. If two processes are either structurally or behaviorally incompatible they cannot collaborate with each other, hence produce a malformed process choreography with erroneous behavior. Structural compatibility considers the structure of interaction. A process choreography is structurally compatible if for each message that is sent there is a participant that can receive it, and for each message that is received there is a participant that can send it. A choreography with such a property is called strong structurally compatible. The weak structural compatibility notion also allows that not all messages that could ever be received need to be sent. However, all messages that can be sent need to be received by other participants [165].

Behavioral compatibility looks at the behavior derived from the interaction of processes in a process choreography. Behavioral compatibility is analyzed with formal analysis techniques and needs a representation with unambiguous semantics like Petri nets or subclasses of Petri nets like Workflow modules or Open nets. The idea is to assure that the sending and receiving of messages in a process choreography is aligned to the internal processes such that all participating processes in a process choreography will not block each other or deadlock due to their interaction.

2.3 BUSINESS PROCESS MODEL AND NOTATION

The Business Process Model and Notation has been released in its current version BPMN2.0 in January 2011 [112]. The Business Process Model and Notation intents to support business process modeling on a wide range of abstraction levels, from the business process level to the more detailed technical implementation level [112]. BPMN has become a de facto standard for Business Process Management [32]. It aims at providing an easy to use graphical modeling notation that is able to express complex business process semantics for both technical and business experts. BPMN consists of different diagram types for modeling both process orchestrations and process choreographies [165]. Each of the diagram types have their particular use case highlighting specific aspect of interest of business processes.

Business process diagrams focus on single process orchestrations whereas collaboration diagrams allow modeling interactions between different process orchestrations. Conversation diagrams portray the interrelation between process orchestrations in a simplified way by reducing them to their core links. They are an informal description of collaboration diagrams and show links between the interacting partners [112]. Choreographies, in contrast to process orchestrations,

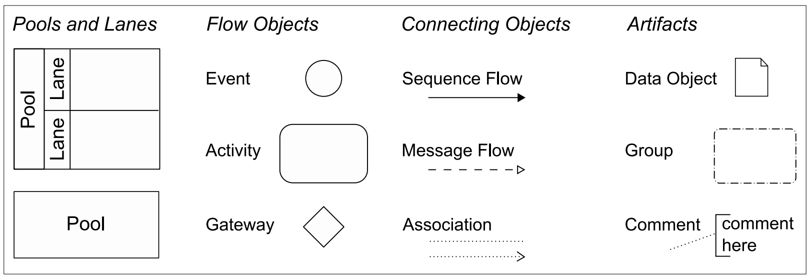


Figure 8: BPMN business process elements [109]

have no central controlling entity. Choreographies can be modeled with collaboration diagrams as well as with process choreography diagrams.

2.3.1 *Business Process Diagrams*

The business process diagram (BPD) is the most commonly used diagram type of the BPMN specification. Business process diagrams are used to describe single business processes as well as business process collaborations. BPDs consist of four main element categories, flow objects, artifacts, connecting objects, and swimlanes [165]. Each category contains different elements which can each have different types. The basic elements are shown in Figure 8 taken from [109].

The flow objects category consists of activities, gateways, and events. Activities depict work that is performed during process execution. Activities can be further specified into tasks, sub-process, or call activity and refined with markers and task types. A task is an atomic activity that cannot be decomposed further. Receive tasks and send tasks represent a communication with a participant [165, 112].

Gateways are used to introduce branching logic into the control flow. Events describe the occurrence of an incident of interest that happens during the execution of a process. An event can be caused by the process itself or can be caused by an external source. In this case the event impacts the business process and needs to be processed [112]. BPMN categorizes events into start, intermediate, and end events. Most of them can be further divided into throwing (outgoing) and catching (incoming) events.

Artifacts represent data objects, i. e., paper forms, electronic information, or physical artifacts, that are manipulated during process execution. These can also be specified as input or output or both of a process activity.

Swimlanes consists of the elements pool and lane. They depict participants, role, and organizational information of a business process. Lanes and sub lanes represent which department or role performs a particular task or activity. Each pool in a process diagram represents a single business process [165, 112].

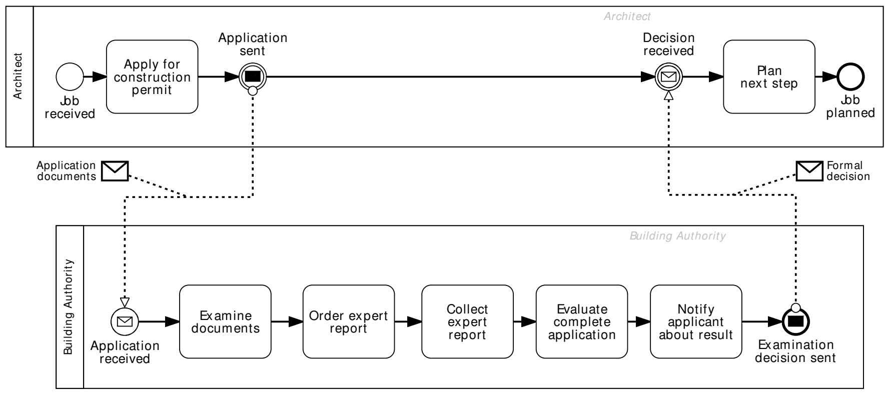


Figure 9: Collaboration diagram

Connecting objects consist of sequence flow and message flow, as well as associations. The sequence flow, often also referred to as control flow, connects flow objects with each other. Message flows can only connect events and activity nodes, as well as pools. A message flow is only used to depict the communication between two communicating processes/pools and not to depict process internal communication [165, 112].

In BPMN2.0 a process collaboration consists of at least two process pools that are interlinked by at least one message flow. Figure 9 shows such a collaboration between the building authority's process and the architect's process. The architect sends his construction permit application to the building authority that evaluates the application and replies with its decision for that case. On a very abstract level, such a collaboration can be expressed by two collapsed pools, however, in this case, no clear statement can be made about the order of the actual execution of the message exchange.

2.3.2 *Choreograhpy Diagrams*

Choreography diagrams were developed to model and represent the behavior between two or more collaborating business processes and depict the message exchange between the different participants [165]. It is an extended type of collaboration diagram [112].

Choreography diagram flow objects are choreography tasks, gateways, and events. A choreography task describes an atomic and synchronous message exchange between two or more participants. A choreography task consists of three parts, the initiator of the message exchange, the task name, and the receiver, respectively receivers of the message. Initiator and receivers are also referred to as participants. A choreography task has always only one initiator [165].

A choreography task can represent only one uni-directional message exchange or a bi-directional request reply message interaction. The connector objects consist only of sequence flows as the choreog-

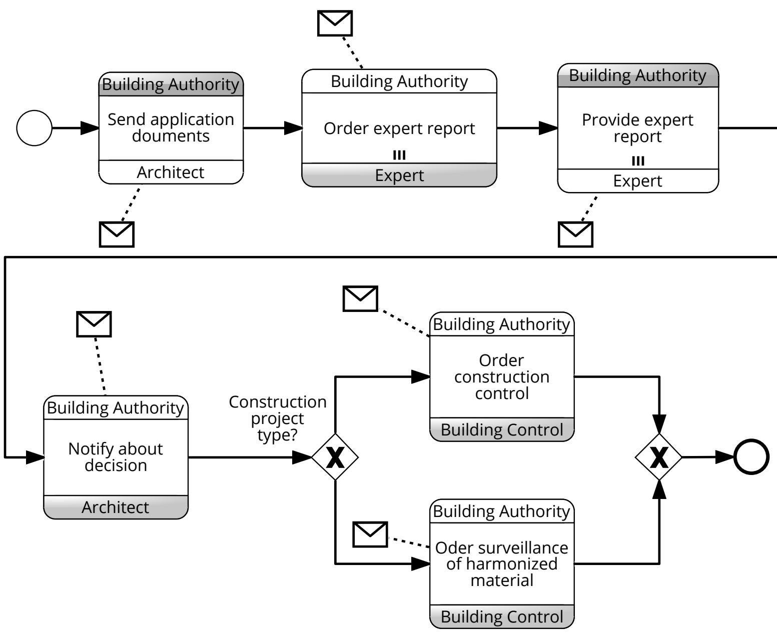


Figure 10: Choreography diagram example

raphy task already describes the message exchange of collaborating process orchestrations. A subset of the start, intermediate, and end events specified for process orchestrations is used in process choreography diagrams. In a choreography diagram exclusive, event-based, inclusive, parallel, and complex gateways are allowed. There are no data artifacts in process choreography diagrams [112]. A choreography diagram can have one or several start events as well as one or several end events [36]. BPMN choreography diagrams need to comply with the enforceability criterion [112]. I. e., the initiator of a task must have participated in the preceding choreography task, else it cannot determine the termination of the preceding task. A choreography that does not comply to this criterion is called non-enforceable [165, 36] Figure 10 shows a choreography diagram depicting message interaction between several actors of a construction permit. First a sequence of choreography tasks are executed that present the message exchange between architect, building authority, and expert. Depending on the kind of construction permit orders for the control of the construction project are transmitted to the building control department.

2.3.3 *Conversation Diagrams*

Conversation diagrams focus rather on highlighting the communication participants of a process collaboration than their behavioral interaction. They provide an aggregation of the messages exchanged between participants in a process choreography [165]. Conversation diagrams consist of conversation elements, participants represented

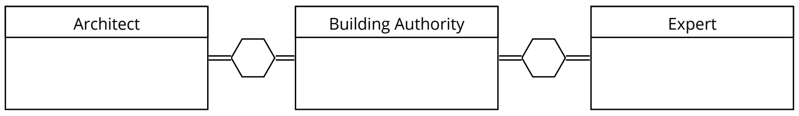


Figure 11: Conversation diagram

by pools, and conversation links that link the participants to a conversation. At least two participants participate in a conversation. Figure 11 shows the elements of a conversation diagram.

2.4 EVENT DRIVEN PROCESS CHAINS

Event-Driven Process Chains (EPC) are a rather informal and simple business process modeling notation. They are one of the most prevalent process modeling languages for business process management, beside BPMN. The focus of EPCs lies on the capturing of business processes and their domain aspects. EPCs cover the modeling of the business process domain in a larger modeling framework, the Architecture of Integrated Information Systems (ARIS) developed by August-Wilhelm Scheer [165, 127].

The EPC modeling notation has only few main modeling elements. The main building blocks are events, functions, connectors, and control flow edges. Additional elements are organization/role, and a group of symbols representing data objects and resources. Figure 12 gives an overview of the EPC elements and their graphical symbols.

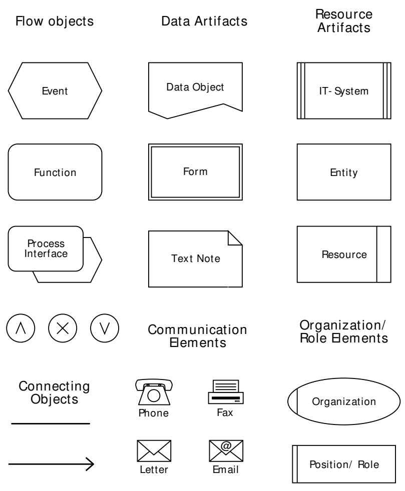


Figure 12: EPC elements

EPC function elements are the equivalent to BPMN tasks and represent a unit of work. Connectors are used to depict the process logic, e. g., decision points or the parallel execution of several functions. Events are passive elements and depict the occurrence of relevant situations [165] .

An EPC must have at least one start and one end event which have only one outgoing and one incoming edge respectively. EPCs are largely bipartite, i. e., functions and events alternate in the process flow. Each function is preceded and succeeded by one event. This usually leads to very large process models [165].

In contrast to BPMN, EPCs do not specify swimlanes and do support modeling choreographies. Roles are attached to function elements representing the execution of the according function by the

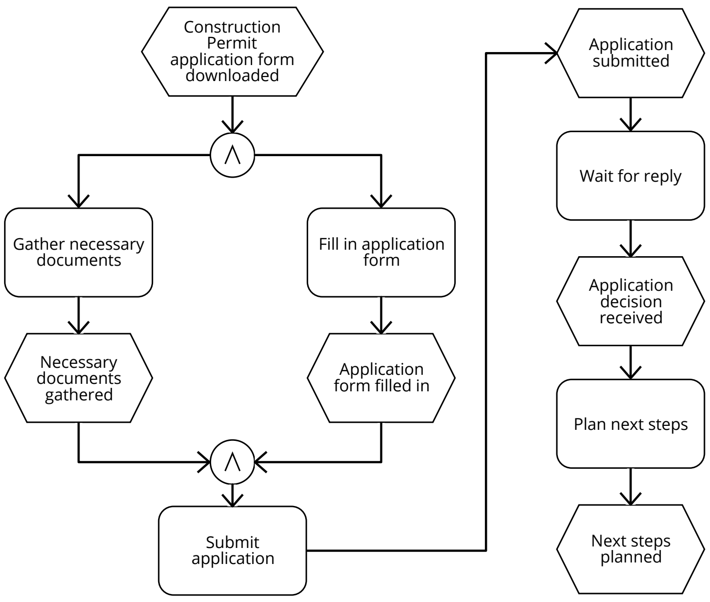

Construction Permit application form downloaded
∧
Gather necessary documents
Fill in application form
Necessary documents gathered
Application form filled in
∧
Submit application
Application submitted
Wait for reply
Application decision received
Plan next steps
Next steps planned

Figure 13: EPC example process

assigned role. Interaction with other processes is difficult to express. In EPCs, there is no specific message flow symbol, hence in many cases the interaction between processes can only be detected by data objects, events, labels of activities, or the process interface symbol. Communication with other processes can be filtered out by analyzing event labels, e. g., when labeled "documents received"'. The process interface elements can be considered special event that instantiates other processes and in this way also expresses an interdependency between processes.

EPCs in general are underspecified and lack well defined syntax and semantics [148]. A construction permit application process modeled in the EPC notation is shown in Figure 13.

2.5 PETRI NETS

Petri nets are a mathematical and graphical modeling formalism developed by Carl Adam Petri. They are used to model dynamic distributed, non-deterministic, concurrent, asynchronous, or stochastic information systems with static structure [110]. Nowadays, Petri nets are widely used in different fields of computer science. In Business Process Management they provide the formal basis for many process modeling languages as they have clear unambiguous semantics [110, 148, 32, 80]. Many powerful workflow analysis techniques were developed on the mathematical foundation of Petri nets.

Petri nets are bipartite directed graphs that consist of transition, places, and directed arcs that connect transitions and places [110, 165].

Definition 11 (Petri net) *A Petri net* PN *is a tuple* (P,T,F), *in which:*

- P *is a finite set of places*
- T *is a finite non-empty set of transitions*
- $P \cap T = \emptyset$ *and* $P \cup T \neq \emptyset$
- $F \subseteq (P \times T) \cup (T \times P)$ *is a finite set of arcs depicting a flow relation*
- $M : P \to \mathbb{N}$ *denotes the marking of a Petri net* (P,T,F) *mapping the set of places onto natural numbers including* 0
- M_s *denotes the initial marking of the Petri net* ◇

A Petri net PN with a designated initial marking M_s is referred to as Petri net system (PN, M_s). For $X = P \cup T$ we denote the preset of a node $x \in X$ as $\bullet x = \{x' \in X | (x', x) \in F\}$ and the postset of a node $x \in X$ as $x\bullet = \{x' \in X | (x, x') \in F\}$.

The static structure of a system is defined by the connected transitions and places. The places that are contained in the preset/postset of a transition t are called input places, respectively output places, of a transition t. A transition t is enabled if all its input places are marked with a token. If an enabled transition fires, it consumes a token from each of its input places and produces a token on each of its output places. I. e., the markings of the corresponding places are changed. In the following we define the firing of a Petri net according to [165].

Definition 12 (Firing of a Petri net) *Let* $PN = (P,T,F)$ *be a Petri net and* M *a marking. The firing of a transition is expressed as state change of the Petri net.*

- $M \xrightarrow{t} M'$ *describes the state change of the Petri net from* M *to* M' *by firing transition* t
- $M \to M'$ *specifies that there is a transition that changes the state of the Petri net, such that* $M \xrightarrow{t} M'$
- $M_1 \xrightarrow{*} M_n$ *specifies that there is a firing sequence* $t_1, t_2, ..., t_{n-1}$ *such that* $M_i \xrightarrow{t_i} M_{i+1}$, *for* $1 \leq i < n$
- *A state* M' *is reachable from state* M *if and only if* $M \xrightarrow{*} M'$ *holds.* ◇

Transitions without any incoming places are called source transitions, transitions without any outgoing places, sink transitions, respectively. Source transitions are able to unconditionally produce new tokes, whereas sink transitions consume tokens without producing new tokens [110]. The dynamic behavior of a Petri net is described by its token play, i. e., its firing sequence. Places can hold several tokens at a time.

Different Petri net classes have been established, e. g., Workflow nets, Place/Transition nets, or Open nets. Place transition nets extend Petri nets by assigning weights to arcs. This allows for example modeling the consumption or production of several resources at once. Graphically this is represented by arcs labeled with natural numbers.

A transition can only fire if its input place is marked by at least the amount of tokens that is assigned as weight to its incoming arc. Similarly, a transition produces the same amount of tokens on its output place that is assigned as weight to the transition's outgoing arc. In the following we introduce the place transition net definition by [110, 165].

Definition 13 (Place transition net) *A place transition net,* (P, T, F, ω)*, is a Petri net with a weighting function* $\omega : F \rightarrow \mathbb{N}$ *that assigns a weight (a natural number) to an arc. Its dynamic behavior is defined as follows:*

- *A transition* t *is enabled if each input place* p *of* t *is marked with at least the amount of tokens assigned by* $\omega((p,t))$*, such that* $M(p) \geq \omega((p,t))$.
- *A firing of a transition* t *removes from each of input place* p *of* t*,* $\omega((p,t))$ *tokens and produces on each output place* p' *of* t*,* $\omega((t,p'))$ *tokens.*
- *The firing of a transition* t *in a state* M *results in state* M' *where* $(\forall p \in \bullet t) M'(p) = M(p) - \omega((p,t)) \wedge (\forall p \in t\bullet) M'(p) = M(p) + \omega((t,p))$. ◇

In the following, properties of Petri nets are introduced, following [155, 148, 165].

Definition 14 (Free-choice) *A Petri net* (P, T, F) *is a free-choice Petri net if and only if for every two transitions* $t_1, t_2 \in T$*, either* $\bullet t_1 = \bullet t_2$ *or* $\bullet t_1 \cap \bullet t_2 = \emptyset$. ◇

Definition 15 (Boundedness) *A Petri net system* (PN, M_s) *is bounded, if there exist an upper bound* $n \in \mathbb{N}$*, i. e., for every reachable state* M' *and every place* p *the number of tokens in* p *is bounded, such that* $M(p) \leq n$. ◇

Definition 16 (Liveness) *A Petri net system* (PN, M_s) *is live iff, for every reachable state* M' *and every transition* t *there is a state* M'' *reachable from* M' *which enables* t. ◇

Definition 17 (Livelock) *A Petri net system* (PN, M_s) *is in a livelock, iff for every reachable state* M' *and a transition* t*, there is a state* M'' *reachable from* M' *which enables* t*, such that* $\exists M', M'' : M' \xrightarrow{t} M''$ ◇

2.5.1 *Workflow nets*

Workflow nets (WF net) are a special class of Petri nets that were designed to represent and analyze workflows with specific structural properties. They pose structural restrictions on Petri nets and the business processes that they represent. The formal semantics of Petri nets allow analyzing the properties of business processes [147, 149].

Structurally, WF nets are restricted by several conditions. A WF net has exactly one initial place that has no incoming flow and exactly one final place that has no outgoing flow. Each transition of a WF net must be on a path from the initial place to the final place. If these conditions hold, the WF net is called structurally sound. In the following we present the definition of WF nets following [165]. It is the basis for the definition of behavioral properties of WF nets.

Definition 18 (Workflow net) *A Petri net* $PN = (P, T, F)$ *is called a Workflow net if the following conditions hold*

- *There is exactly one initial place* $p_i \in P$ *that has no incoming flow, such that* $\bullet p_i = \emptyset$*.*
- *There is exactly one final place* $p_o \in P$ *that has no outgoing flow, such that* $p_o \bullet = \emptyset$*.*
- *Each transition* $t \in T$ *and every place* $p \in P$ *of* PN *is on a path from* p_i *to* p_o ◇

A Petri net system where PN is a WF net is called a workflow system. The initial marking of a WF net marks only the initial place with a token, all other places are unmarked. In this context the token represents an instance of the WF net that is started with the start of the token play. In the final marking only the final place of the WF net is marked, and all other places are unmarked. Based on this formalism a range of structural and behavioral properties for WF net have been defined [165].

The behavioral property soundness describes the ability of a WF net to always reach its final place and not deadlock on any possible path. Furthermore none of its transitions is dead, i. e., each transition will be fired on at least one execution path. From any state M reachable from the initial state, the final state M_Ω is reachable where only the final place is marked and any other place of the net is unmarked. This means that every process instance of the represented process terminates in a proper way [147, 149, 165]. The soundness criterion is summarized in the following definition according to [165].

Definition 19 (Workflow net states) *Let* $PN = (P, T, F)$ *be a workflow net,* $p_i \in P$ *be its initial place,* $p_o \in P$ *its final place, and* M, M' *markings.*

- M_s *is the start state in which only* $p_i \in P$ *carries a token and all other places are unmarked*

- M_Ω *is the final state in which* $p_o \in P$, *the final place, is marked with exactly one token and every other place is unmarked*
- $M \geq M'$ *if and only if* $M(p) \geq M'(p), \forall p \in P$
- $M > M'$ *if and only if* $M \geq M' \wedge \exists p \in P : M(p) > M'(p)$

Definition 20 (Soundness) *A workflow system* (PN, M_s) *with a WF net* $PN = (P, T, F)$ *is sound if and only if,*

- *for every state* M *reachable from the initial state* M_s *there is a firing sequence, leading from* M *to the final state* M_Ω*, such that* $\forall M(M_s \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} M_\Omega)$
- M_Ω *is the only state reachable from* M_s *with at least one token in place* p_o*, so that* $\forall M(M_s \xrightarrow{*} M \wedge M \geq M_\Omega) \Rightarrow (M = M_\Omega)$
- *there are no dead transitions in the workflow net in state* M_s*, so that* $(\forall t \in T)\exists M, M' : M_s \xrightarrow{*} M \xrightarrow{t} M'$ ⋄

The notion of soundness is strict in regard to behavioral and structural properties of business processes. In the real-world only few of the business processes will comply to this property. In this regard a less strict and weaker notion of soundness, relaxed soundness, has been introduced in [30]. Relaxed soundness allows for deadlocks or improper termination in some execution paths, however each transition must be on a sound execution paths. This means that each transition is on at least one firing sequence reached from the initial state M_s from which it can reach the final state M_Ω [30, 156, 165].

To define relaxed soundness we introduce the notion of a sound firing sequence according to [165].

Definition 21 (Sound firing sequence) *Let* (PN, M_s) *be a workflow system. Let* δ, δ' *be firing sequences and let* M, M' *be states.* δ *is a sound firing sequence if it leads to a state from which the final state* M_Ω *can be reached:* $M_s \xrightarrow{\delta} M$ *and* $\exists \delta'$ *such that* $M \xrightarrow{\delta'} M_\Omega$

Definition 22 (Relaxed soundness) *A workflow system* (PN, M_s) *is called relaxed sound, if and only if, each transition of* $PN = (P, T, F)$ *is an element of some sound firing sequence, i. e., reaches the final state* M_Ω

- $\forall t \in T\ \exists M, M' : (M_s \xrightarrow{*} M \xrightarrow{t} M' \xrightarrow{*} M_\Omega)$ ⋄

2.5.2 *Open nets*

Open nets [92, 94] were developed to model and analyze the compatibility and interaction of interorganizational workflows and collaborating concurrent systems, web services or processes, e. g., in Service Oriented Architectures (SOA). They extend Petri nets with interface places and a set of final markings. The set of places is divided in disjoint sets of internal, incoming, and outgoing places. Incoming and outgoing places are interface places that depict their interface to the external environment, systems, or in our case other business processes [92, 94]. Through their interface the communication and interaction with external partners takes place. Each Open net has an initial marking that defines the initial state of the Open net as well as a final marking that defines the termination of an Open net [92, 94]. In the initial marking the interface places are not allowed to be marked. The firing rules of Petri nets are equally valid for Open nets.

For a set Z we denote with $MS : Z \to \mathbb{N}$ the multiset over Z, where each element of Z can occur multiple times (i. e., $z \in Z$ occurs $MS(z)$ times). We write multisets as a formal sum of their elements, e. g., $2 \cdot p_1 + p_2$ for the multiset containing two exemplars of p_1 and one of p_2. The empty multiset is denoted as 0.

Workflow modules are a restricted form of Open nets that are presented in Section 2.5.3 particularly tailored to the BPM domain. Following [95, 92] we define Open nets as follows.

Definition 23 (Open net [95, 92].) *An Open net is a tuple* $O = (P, T, F, M_s, \Omega)$*, in which:*

- P *is a finite set of places that is partitioned into pairwise disjoint sets of internal places* P^N*, input places* P^I*, and output places* P^O
- $P^{IO} = P^I \cup P^O$ *denotes the set of interface places*
- T *is a finite set of transitions, disjoint with* P
- $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ *is the flow relation assigning weights to arcs*
- *A marking* $M : P \to \mathbb{N}$ *is a multiset over* P*, i. e., it assigns to each place the number of tokens on this place,* M_0 *denotes the initial marking of the net,* Ω *the set of final markings*
- $\bullet t$ *is called the preset,* $t\bullet$ *the postset of a transition* $t \in T$*. Both are multisets over* P*and denote the amount of token consumed from, respectively, put on a place.*
- $t \in T$ *is activated in a marking* M*, denoted by* $M \xrightarrow{t}$ *if* $\forall p \in P : M(p) \geq \bullet t(p)$*, i. e., if there are enough token on* p *for* t *to consume*
- *Firing an activated transition* t *leads to a follower marking* M' *defined by* $M' = M - \bullet t + t\bullet$ ◇

The introduction of desired final markings allows for a clear definition of deadlocks.

Definition 24 (Deadlock) *A Petri net system* (PN, M_s) *is in a deadlock if it is in a state* M' *with a marking from which no other marking is reachable, and* M' *is not the final marking, i. e.,* $M' \notin \Omega$ ⋄

Two Open nets O_1, O_2 have shared interface places if their interface places match. An interface place that is not shared between two Open nets is called free. The formal definition is as follows according to [92].

Definition 25 (Shared, free interface place [92].) *A interface place* $p \in P_1^{IO} \cup P_2^{IO}$ *of two Open nets* O_1, O_2 *is* shared *between* O_1 *and* O_2 *if* $p \in P_1^{IO} \cap P_2^{IO}$*. If this condition does not hold,* p *is* free *between* O_1, O_2*.* ⋄

Two Open nets O_1, O_2 can be composed if they are interface compatible. This is the case, if a place p is shared between two Open nets O_1 and O_2, then p is either an output interface of O_1 and an input interface of O_2, or vice versa [92]. The formal definition of interface compatibility according to [92] is given in the following.

Definition 26 (Interface compatibility [92].) *Two Open nets* O_1 *and* O_2 *are called* interface compatible *if each shared interface place is an input place of one Open net and an output place of the other net, i. e.,*

- $(P_1^{IO} \cap P_2^{IO}) = (P_1^I \cap P_2^O) \cup (P_2^I \cap P_1^O)$ ⋄

The composition of two Open nets O_1, O_2 is realized by fusing the matching input and output places of both nets O_1 and O_2, and results in a new composed net O. Fused interface places become internal places in the composed net [95, 92].

Definition 27 (Composition of Open nets [92].) *The composition of two interface compatible Open nets* O_1, O_2 *results into a composed net* $O_1 \oplus O_2 = (P, T, F, M_s, \Omega)$ *with*

- $P^I = (P_1^I \setminus P_2^O) \cup (P_2^I \setminus P_1^O)$
- $P^O = (P_1^O \setminus P_2^I) \cup (P_2^O \setminus P_1^I)$
- $P^N = P_1^N \cup P_2^N \cup (P_1^I \cap P_2^O) \cup (P_2^I \cap P_1^O)$
- $T = T_1 \cup T_2$
- $M_s = M_{s,1} + M_{s,2}$
- $\Omega = \{M_1 + M_2 \,|\, M_1 \in \Omega_1 \wedge M_2 \in \Omega_2\}$
- $F_1 \cup F_2$ ⋄

2.5.3 *Workflow modules*

Workflow modules were proposed to model inter-organizational business process interaction and assess their interoperability [90]. A workflow module is an extended Workflow net and a special class of Open nets. It has special communication places (input places, output places) that allow for composition with other workflow modules representing their interaction. Each workflow module represents a business process.

The composition of workflow modules is analogous to the composition of Open nets described in Section 2.5.2. Two workflow modules can be composed if their output and input communication places match and their internal transition and internal places are disjoint. This means that each output or input place of one workflow module needs to match the according input or output place of the other workflow module. If this is the case, they are called syntactically compatible. When composed, the communication places become internal places. The new composed net is converted into a structurally sound Workflow net by adding a global initial and a global final place and two transitions that connect them to the according initial and final places of the composed workflow modules. The composed Workflow net can then be checked for soundness. The notion of soundness appeared too restrictive for composed web services, so that Martens [90] introduced the notion of weak soundness for process choreographies.

Definition 28 (Weak Soundness [165, 90]) *Let* (PN, M_s) *be a workflow system with initial state* M_s*. A workflow system is weak sound, if and only if, the following holds:*

- *For every state* M *reachable from the initial state* M_s*, there is a firing sequence from* M *that reaches the final state* M_Ω*, i. e.,* $\forall M(M_s \xrightarrow{*} M) \Rightarrow (M \xrightarrow{*} M_\Omega)$
- M_Ω *is the only state reachable from state* M_s *with at least one token in place* p_o*, i. e.,* $\forall M(M_s \xrightarrow{*} M \wedge M \geq M_\Omega) \Rightarrow (M = M_\Omega)$ ◇

Weak soundness allows for dead transitions in the composed Workflow net, however the final place must be reachable from all reachable states M in the net. A weak sound Workflow net is free from deadlocks and will terminate properly.

The notion of usability of a workflow module describes that the composition with another workflow module results in a weak sound composed system and that there exists at least one such workflow module. A composed system that is weak sound assures the absence of serious errors [90, 91] Based on the usability notion, Martens defines the notion of semantic compatibility for two workflow modules. Two workflow modules are semantically compatible if their composed system is usable [90, 91].

2.6 FORMAL ANALYSIS

Formal analysis is a means for the verification of information systems. Information systems are modeled and analyzed with mathematical concepts, e. g., Petri nets. For this, properties of the system under examination are defined in a formal way. Based on the underlying mathematics these systems are verified for their correctness according to the defined properties.

The idea is to be able to make a statement of the characteristics of systems without the need to verify the real-world system, which in most cases is not even possible. Baier and Katoen [6] state that formal verification is "applied mathematics for modeling and analyzing ICT systems". The aim of formal analysis is to create correct information systems.

In the BPM domain, especially in regard to business processes, business process choreographies, and business process interdependencies, the formal properties of interest are of structural and behavioral nature. Structural properties refer to the composition of the model, e. g., which elements, types are used, and how they are connected. Many business process modeling languages define syntactical properties that specify containment or connection relations between elements. Based on the specification process models can the be verified for their compliance to these syntactical properties. E. g., Workflow nets have exactly one initial and exactly one final place. A net with two initial places would not meet the required structural property and not qualify as a Workflow net.

In regard to business processes the behavior of a business process is depicted by all its execution paths that are defined by the process model. Correct behavior is defined by desired behavioral properties, e. g., soundness, which assures proper termination of all process instances of a process model. Such formal properties allow for their verification with a model checker.

The most common approach for analysis of business processes is to use a mapping to Petri nets and their subclasses [80]. Petri nets have clear execution semantics and provide strong mathematical theory for process analysis. Structural properties of Petri nets are structural soundness for Workflow nets, and interface compatibility for Open nets and workflow modules. Behavioral properties for different classes of Petri nets are deadlock freedom, livelock freedom, boundedness, freedom of dead transitions, soundness, relaxed soundness, and weak soundness.

A correct structure of a process models is often largely assured by business process modeling tools. Although it is assumed that a process model should be free of deadlocks and livelocks [36], only few business process modeling tools allow for the verification of behavioral properties such as deadlocks or livelocks.

Behavioral properties can be expressed as a reachability problem, in which it is to decide whether a desired state can be reached. Such a state is described by a particular marking of the Petri net system.

Due to their complexity some verification problems expressing desired properties cannot be checked with Petri nets. The behavioral analysis of large Petri nets may run into computational problems due to state space explosion [50, 145]. Due to the state space explosion problem, it is often necessary to abstract from irrelevant details when modeling the system of interest. E. g., the restrictions of Workflow nets create a formalism that allows for behavioral analysis also for larger process models, even though the state space explosion problem still exists [145, 49].

The reachability graph functions as the starting point for the analysis of a Petri net, given that there is only a finite amount of states. In general a reachability graph can be infinitely large. To overcome the problem of infinite states the concept of coverability graphs was introduced. A coverability graph is finite and approximates the according reachability graph. Some of the reachability problems for Petri nets with infinite states are decidable with the help of the coverability graph. The coverability graph is used to determine boundedness of a Petri net and its places, to find dead transitions or analyze different kinds of soundness [65, 131, 168, 156].

All of the aforementioned verification tasks of behavioral properties can be considered model checking problems. Model checking problems are specified in temporal logic to find out how a system evolves over time [19]. The desired behavioral problems are expressed in mathematical models, in most cases state predicates, Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) formulae, that map them to a reachability or coverability problem. This allows finding incorrect or inconsistent behavior of the systems under examination. Temporal logic problems are classified by the underlying assumptions that time has a linear (LTL) or branching structure (CTL) respectively [19].

The complexity of model checking is polynomial in size of the model determined and in lengths of its specification in temporal logic [19]. Optimizations of the computation resulted into a linear complexity in regard to the product of the length of the formula that describes the desired state and the size of the state transition graph [19]. As large Petri nets easily suffer from state space explosion, the usability of model checking is restricted to smaller Petri net models. The complexity of solving LTL problems is PSPACE-complete. Although it is exponential in length of the formulas, it was found that it is linear in the size of the global state space. Hence it is only applicable for short formulas [19].

The tool LoLA (A Low Level Petri Net Analyser), a well-established Petri net verification tool, allows solving problems that have practical relevance due to its many optimized state space reduction techniques [132, 167]. According to Wolf [167] and Lohmann [78] presenting different use cases, LoLA can tackle model checking problems with a state space of about ten million states.

2.7 SUMMARY

This chapter introduced the formal foundations and concepts for the specification and description of our Business Process Architecture approach. The concepts of business process orchestrations and choreographies in Section 2.1 and Section 2.2 as well as the introduction of different business process modeling notations in Section 2.3 and Section 2.4 described the elements that are structured by PAs. These concepts and the presented process model decomposition techniques are used in Chapter 7. Section 2.5 and Section 2.6 introduced Petri nets and the formal foundation for the composition, analysis and verification of workflows. We rely on these concepts in Chapter 5 and Chapter 6 for providing clear semantics of our Business Process Architecture approach and its analysis.

3 RELATED WORK

This chapter is based on the published research papers [42, 40, 38, 45, 43, 39]. This chapter describes the state of the art of PA research and other related research field. It puts our research into the overall BPM context. We present research on process model repositories and enterprise architecture as larger setting of PAs. Beside the current research in the PA field, we introduce and examine research on business model abstraction and business process choreographies. The evaluation of the related work will result into an overview of existing research and research gaps. The identified research gaps will serve as foundation, motivation, and basis of requirements for our PA approach, which will be introduced in succeeding Chapters.

Business process architectures have received new interest in BPM research. This topic cannot be solely attributed to one BPM research area. As PAs are used for structuring and managing process models in process model repositories. research on process model repositories specifies the coarse environment of PA research. Depending on the business process architecture approaches chosen, PAs describe hierarchical and horizontal interdependencies between processes. In regard to horizontal interdependencies, they relate in particular areas to research on process choreographies. PAs provide an abstract view on process models. Consequently, our work touches the work on process model abstraction. In this chapter we introduce those different research fields and elaborate on existing approaches and techniques. As a result, we gather the main characteristics of the different approaches and summarize them in an evaluation in Section 3.5. We highlight gaps in current research that we aim to fill with our BPA approach.

3.1 PROCESS MODEL REPOSITORIES

With the dispersion of BPM in companies and organizations, these organizations collect hundreds or thousands of process models [125, 1]. In this regard the need for the effective management of large process model collections in process model repositories arose. Bernstein states, a *repository* is a "shared data base of information ... artifacts used by an enterprise" [13] and is considered to be the centerpiece to integrate tools that leverage the stored information. Process model repositories provide structured access to stored process models and support process specific tasks, e. g., life cycle management, [171]. A lot of research on process model repositories deals with development of functionality used for effective process model management. As PA approaches are part of process model management, process model repositories provide the context and environment for PA research.

Process model repositories address a variety of users that range from various kinds of process model experts to employees of a company that only study process models for learning work procedures and follow internal or external regulations. Domain and process modeling experts create, reuse, and manage process models in large collections [125, 1], or business analysts examine process performance.

A large body of research discusses specific aspects of process model repositories, e. g., structured query search [4, 12], a unifying process meta model for stored processes [75], and automatic support for the business process lifecycle [82]. Shahzad et al. [134] and Yan et al. [170] conducted surveys and derived requirements of general process model repositories. In particular, Yan and Grefen [169] define a framework that captures aspects of a business process model repository, i. e., process data, process functions, and process management.

While many repositories are specific to one certain process format [170], other repositories like Apromore[1] can deal with several process modeling notations [75]. During analysis of process model repositories, we encountered many different process modeling languages, common ones, e. g., EPC [66] or BPMN [112], but also proprietary and rather informal notations, e. g., PICTURE [11] and FAMOS[2]. Hence, a PA approach should be modeling notation independent.

Although different notations, their meta data, or the process model themselves provide raw information on their interdependencies, this information is not used for generating additional information. A large share of information is contained in the inscription of modeling elements but not put into a larger picture or overview. Dijkman et al. [34] introduce current challenges for managing large process model collections. Among others, they highlight the need for organizing process models in a process model collection and provide overviews of their interdependencies.

A large part of users of a process repository has problems of finding process models of interest due to cluttered lists, overviews, and limited process model access possibilities. Referring to modeling notations, Patig et al. [114] emphasize the need for human centered languages and tools. This also includes the need to reduce complexity by providing overviews and easy access to the process models in a process model collection.

These challenges are dealt with by a relatively young topic of process model management, business process architectures, that aims at structuring and organizing process models within a process model

1 Advanced Process Model Repository – http://apromore.org

2 FAMOS Modeling Handbook - Accessed: 6th June 2014 http://www.d-nrw.de/fileadmin/user_upload/d-NRW_Dateien/KDV/modellierungshandbuch_2_2011-03-17.pdf

collection [34]. The structure given by a PA is the basis for providing navigation structures for a process model repository which is one of the main approaches to access, use, and reuse process models in a business process model repository, apart from meaningful search capabilities. In this regard, our work adds to the field of process model management in process repositories by providing a new PA methodology that can be applied to large process model collections. Recent research on PAs aims at providing new perspectives on process models that serve particular purposes; also addressing automatic techniques to derive such structures [37].

Most of the known process model collections in research contain PAs omitting process interdependencies. One of the most commonly used collections is the SAP reference model [21], published in 1997, that comprises 604 EPC process diagrams that describe the SAP R/3 system. The process models in the process collection are grouped along functional lines in a hierarchy up to three directory layers.

IBM released a collection of 735 business process models from the Websphere process modeler available as BPMN process models, which cover various industrial domains, e. g., finance and telecommunications, [52]. The process models in the IBM model collection are categorized according to their domains.

In the course of the BPM academic initiative[3] (BPM AI), a set of *29825* process models (August 2012) created by students in various process modeling languages has been made available to the research community [71]. The BPM AI does neither contain business process architectures nor any kind of categorization of its process models as the process models originate from student tasks.

From quantitative analysis of many process models, one can draw conclusions that influence the way experts are using process models. Related work [107, 71] showed that the majority of models uses only few distinct modeling constructs and is of low complexity, which can guide in reducing the spectrum of modeling languages and hence, ease and accelerate process model design. Guidelines towards process modeling [97, 101] and labeling [100] show how to reduce error proneness and increase model understanding. These aspects highlight the need to reduce the complexity of process models and their interdependencies for users and facilitate the better understanding of process context.

3.2 BUSINESS PROCESS ARCHITECTURE APPROACHES

Recently, empirical studies [35, 84, 85, 33] on the use of PAs in practice have been undertaken that show that there are many challenges in regard to defining, applying, and evaluating PAs. Several works deal with frameworks for classifying [3, 54, 56, 53, 86] and evaluating PA

3 http://bpmai.org

approaches, e. g., [54, 56, 53]. In few papers guidelines for constructing PAs are clearly outlined, e. g., [31, 106, 57, 55]. In the following we will present the different topics of PA research and outline PA approaches.

ENTERPRISE ARCHITECTURE APPROACHES. Enterprise architecture frameworks describe the relations between the IT-system model, the business model, the business process model, and other parts of the enterprise architecture. Business process architectures are often mentioned as part of enterprise architectures methodologies but in most cases without clear description of the underlying PA concept.

Incorporated into a larger organizational enterprise architecture framework different business process architecture methodologies have been proposed earlier, e. g., in Zachmann [174] or in Scheer et al. [128].

In one of his early works, Zachman [174] designed an information systems architecture for organizations. According to him there are several views necessary for modeling an organization. His rather informal enterprise architecture framework encompasses several domains, among others he also looks at structuring business processes along functional lines.

The ARIS architecture and reference models for business process management is a widely used framework [128, 127, 129, 22]. It is a multi-layer enterprise architecture framework that consists of four different levels; process engineering, process planning and control, workflow control, and application systems. There are means to interconnect elements from different layers, however, the framework does not provide concrete guidelines on how to model such interconnections nor a formal underpinning that allows defining clear relations. In [129] Scheer et al. present a reference model based PA approach that groups processes along business functions. On process model level they use EPCs that by design have a single process model focus. The interrelations to other process models are symbolized by a process interface and process call element.

Davis [22] provides a practical guide on the advanced use of the ARIS design platform. ARIS envisages a hierarchical overview of a process model that is based on its functions and the tasks of their sub-processes. In regard to structuring a whole process collection Davis [22] proposes a five level based process hierarchy similar to Scheer et al. [128, 127, 129]. On first level, he situates the enterprise map consisting of value chain diagrams. The creation of value chains is not described and appears to be a rather creative task or a standard scheme from the manufacturing area which does not suit modern company structures [59, 14]. On second level, he puts process area maps, on third level key process models, that are decomposed into more detailed process models on level four, and on level five he depicts activity models. The elements of the different layers can be referenced with "consists of" or "relates to" concepts that hierarchically or horizontally link elements together but do not provide information on the influence of those relations.

FRAMEWORKS AND SURVEYS. A large set of PA literature are surveys on PA approaches and frameworks to classify and evaluate those approaches. The surveys by Dijkman et al. [35, 33] and Fettke et al. [54, 53] examine mainly PA approaches developed in research whereas the surveys by Malinova and Mendling [84] and Malinova et al. [85] evaluate and examine the use of PAs in practice.

Malinova et al.[85] performed an empirical study on the use of PAs in companies and propose a classification for PA types based on their findings. All approaches evaluated by Malinova et al. are based on the classification of single processes. They identified four PA types: hierarchical, pipeline, divisional, and service-oriented. Most of the organizations had a business process architecture of three levels. They note that the design of PAs is rather complex and context dependent. PAs were used in the re-/design and analysis, and evaluation phase of the BPM lifecycle. Depending on the structure of the project PAs were used top-down or bottom-up.

Malinova and Mendling [84] provide an overview on process map approaches and their effects on the success of business process management projects. For this they surveyed 15 process maps from practice. The term process maps refers to one of the top layers of a business process architecture but it is also often used as synonym for PA. According to Malinova and Mendling current PA/Process map approaches lack informational power and do not provide any beneficial effect on process management success as they are difficult to interpret and not directly connected to goals or processes. They state that in many cases there is no consistency between the different layers of a process model collection, e. g., business process and process map layer. Only few PA or process maps show interdependencies between processes and give a good overview on the process model collection and its informational treasure [84]. These findings coincide with the type of examined PAs from practice in [85].

Green and Ould [56] and Fettke and Loos [53] present frameworks for evaluating and classifying PA approaches. Green and Ould [56] highlight that most PA approaches are used for organizing the elicitation of process models top-down. As a first step of a process model elicitation project a PA is designed and then the subsequent business process models. Their framework is based on a questionnaire covering several dimensions for evaluating PAs. The dimensions are form view, content view, purpose view, and lifecycle view. These dimensions represent a range of requirements posed on PA approaches. The form view dimension refers to well-defined PA and process models in regard to syntax and semantics. The content view dimension examines if there exists an underlying theoretical framework for the PA, a clear specification of relationship types between processes, or comprehensibility of the PA to experts and non-experts among other questions. Aspects like the identification of processes, facilitation of communication, and overall utility are covered by the purpose view dimension. The lifecycle view deals with the ease of use of the PA approach, ease of maintenance, effectiveness and efficiency, among others. Green and Ould [56] criticize that in many cases process modeling is performed only piece by piece without a complete overview which is error-prone. They highlight the necessity for a coherent modeling of business process and for visualizing interdependencies between processes. This supports the analysis of process interdependencies and in the same time helps to assure completeness and avoid errors due to missing processes

Fettke and Loos [53] develop a classification system and classify 26 reference information model approaches including business processes. They present four general types of classification; basic classification, hierarchical classification, faceted classification, and characteristic-based classification. Basic classification allows each classification object to be a member of one class only. Hierarchical classification orders classes hierarchically. One class can encompass several sub-classes.

Faceted and characteristic-based classification are multi-dimensional classifications [53]. Faceted classification classifies objects according to a set of attributes, the facets. A classification object is classified with multiple attributes that allows for retrieving it from different paths. Characteristic-based classification classifies each classification object according to several attributes. Facets are mutually exclusive whereas characteristics are not.

The classification system types are also often found in PA approaches whereas the classification attributes differ according to the methodology chosen, e. g., goal-oriented or departmental [35, 33]. Process models are grouped along one or several characteristics. Fettke and Loos evaluate the classifications of information models, however, do not provide a method to design PAs [53].

In a subsequent work, Fettke and Loos [54] examined 30 different reference models approaches. Their understanding of a reference model refers to a complete information model or process model collection like the SAP reference model or the structure of a process model collection like the APQC [3]. Their evaluation framework consists of three main categories: general characterization, construction, and application. The relevant categories relating to PA aspects are construction and application. The construction category consists of aspects like existence of a modeling framework, construction method, or evaluation methods among other aspects like number of process models and modeling language. Use cases, reuse, and customization as well as application method examine the concrete application area of the reference model. Their categories show that a clear construction method with well-defined syntax and semantics as well as applicability are desired aspects for PA approaches. According to Fettke and Loos [54] only few approaches describe how to create an own model.

Koliadis et al. [70] raise a set of 22 questions to evaluate functional aspects of ten different business model and enterprise architecture approaches. Their evaluation includes approaches on the strategic organizational level like value chains (VC), strategy maps (SM), business motivation model (BMM), the I*-framework, e3-Value (e3) or the ARIS House of Business Engineering (HOBE). Their evaluation criteria encompasses questions on enterprise structure, enterprise motivation, enterprise capability, enterprise relationships, and enterprise risk. They examine the structure of goals and their relation to business processes, the actors performing operations, their processes in the company and the relationships between actors. Especially, the enterprise relationship category is used to evaluate the type of relationship between actors and how strong the dependency between actors is. The risk category focuses on critical and vulnerable actors, processes, and relationships. These questions show the relevance of processes, actors, and involved resources like data for PAs.

Most approaches showed deficiencies in meeting the requirements for PAs specified by the evaluation questions, especially in regard to the interdependencies between enterprise actors and processes. Many of the evaluated frameworks are strong in identifying key enterprise actors, the enterprise goals, defining performance objective and relating actors to processes. The actor focus suggests that many of those frameworks group processes along organizational lines.

Different kinds of PAs have been examined in regard to their usefulness and use in an extensive survey with 39 practitioners by Dijkman et. al [33, 35] who identify five types of PA approaches: goal-based, action-based, reference-based, function-based, and object-based. They noted that most practitioners prefer to use a reference model-based and function-based PA approach. Practitioners also relied on a combination of approaches rather than choosing a single one. In regard to identify popular combinations of PA approaches they studied four use cases from practice. In three out of four of those use cases object- and function-based approaches were used in combination. The most common relations observed were decomposition and ordering. Dijkman et al. assume that ordering relations referring to temporal relations appear to play an important role although not being present in literature [35]. Based on their findings they propose a PA framework based on multifaceted classification. It organizes processes along the hierarchical decomposition of business functions and the classification along permanent business objects [35]. The PA use cases they examined also contained sequence ordering relations but the visualization of end-to-end processes or a clear process execution order is not part of the approaches examined in [35].

Maddern et al. [83] examine current end-to-end process modeling approaches to assess the current state of research in this field. End-to-end process modeling identifies process chains that serve the customer from the moment of contact to delivery of a service or a product. The idea shared by end-to-end process modeling approaches consists of providing a customer centered and holistic overview on the processes involved into the delivery of a service or production of a product. Maddern et al. [83] state that the end-to-end term is not clearly specified in research and practice. End-to-end modeling comes with many challenges, e. g., defining the scope of an end-to-end process. The main challenges include providing a holistic approach over a set of processes, the difficulty of setting boundaries of an end-to-end process, and the management of end-to-end processes across department boundaries. They call especially for a shift from linear functional approaches to a holistic approach considering interdependencies based on input/output flows between all processes and their resources [83]. Having a holistic scope, with end-to-end view and all involved internal business processes, our BPA approach will help to close this gap.

BUSINESS PROCESS ARCHITECTURES BASED ON CLASSIFICATION. Business process architecture approaches that rely on classification systems along functional [130, 31, 84], organizational, object, goal [2, 58], or activity lines [106] are most commonly found in BPM literature. Many employ a hierarchical ordering and classify single process models ignoring process interdependencies. A large body of work [16, 60, 86, 17, 64, 57, 67, 8, 106, 85, 31, 96] falls into that category.

Functional and hierarchical PA approaches from practice are described in [58, 59, 22, 85, 3]. Harmon [59] describes the urge of companies to create PAs in combination with defining metrics to measure process success. For this, processes are related to process goals and balanced scorecards. Balanced scorecards require PAs based on goals and functions [58].

The American Productivity & Quality Center (APQC) and its members developed the APQC's process classification framework, a taxonomy of cross-functional business processes [3]. The APQC classifies business processes on top level into operating processes, and management and support services. In total there are 12 enterprise level categories, 5 categories in the main category operating processes and 7 categories in the category management processes and support services. The process classification is structured hierarchically into five levels; 1. category, 2. process group, 3. process, 4. activity, and 5. task. It was developed as an open standard to support process management and improvement as well as benchmarking processes within and among organizations regardless of industry and size [3].

In a similar attempt Malone et al. [86] created the MIT Process handbook that classifies a range of processes along the two dimensions part of business process and type of business process. The processes and their activities are hierarchically classified using *part of*, *specialization*, *generalization*, and *uses* relationships. It covers several business model domains ranging from manufacturer, wholesaler to HR broker or financial trader among others.

The Supply Chain Council Inc. (SCC), a global non-profit organization, developed the Supply Chain Operations Reference model (SCOR) [133]. The SCOR model shows the consortium's view on supply chain management and consists of six primary management processes: plan, source, make, deliver, return, and enable. The SCOR model has a hierarchy of four levels; process types (scope) on top level, process categories (configuration) on second level, process elements (steps) on third level, and activities (implementation) on the fourth level. The framework encompasses metrics, and best practices to measure and improve business processes. Similar to APQC the SCOR model intends to provide a framework to evaluate and compare supply chain activities and performance [133].

As one of the few approaches, Dijkman [31] proposes concrete PA guidelines to create PAs. He provides guidelines to identify and hierarchically decompose business processes based on business functions and case type.

PRODUCT BASED PA. Moreira and Fillies [105] classify a public organization's products and services according to a product classification framework, the ICT class model of the public sector that provides ten classes in three categories; information, communication, and transaction. Their classification is part of a business process model analysis architecture. They look at interfaces by considering the information flows between activities that represent the interchange.

Castellanos and Correal [18] consider PAs as part of an enterprise architecture. Their aim is to align the information system architecture elements with business process elements from the business process architecture. To align their business processes with data objects from the information system architecture they use an ontology to map both architecture elements. Processes that use the same data object are classified in the same category. The interdependencies between business processes based on data objects are not considered in their mapping.

REFERENCE BASED PA. PA approaches in [127, 53, 129, 67, 54, 63, 2] use a reference model to classify their process models.

Klein and Petti [67] present a method to use the MIT process handbook to model and redesign value chains based on building blocks.

Their idea is to recombine building blocks to create new process models. Their classification system refers to a classification by name. For the redesign of processes they use the hierarchy, inheritance, and specialization concepts of the process handbook to find alternatives for modeling the "to-be" process model. The main focus of their work is the re-design of process models and not the structuring of a process model collection.

Anderson et al. [2] aim at finding a good matching reference model from best practice for re-engineering processes under evaluation. Their approach requires process models to be organized along goals. For determining a best fit between the process to re-engineer and the best practice reference model they rely on the associated goals.

HORIZONTAL INTERDEPENDENCIES AND MULTI-PROCESS BUSINESS PROCESS ARCHITECTURE APPROACHES. Only few approaches [76, 55, 56, 57, 63, 126] consider horizontal process interdependencies in their methodology and consider them when constructing a PA.

Designed by Green and Ould, the RIVA approach is presented and evaluated in [55, 56, 57]. The RIVA methodology is based on business entities (essential and designed) and creates a business process architecture based on interdependencies between processes but also considers organizational interaction. It specifies the relationships *encapsulates*, *interacts with*, and *activates* between processes. Relationships between roles and organizations can be defined as well. Looking at different kinds of interdependencies between processes there is no formal syntax or defined semantics for their concepts. By visualizing the interdependencies between processes they can be analyzed manually.

Jacobs et al. [63] present an enterprise architecture approach based on a transfer of a data warehouse techniques to the enterprise architecture model domain. They aim at providing different views on a company's operations and IT-systems by mining different reference models. In this way they create a PA that allows slicing and dicing process models and creating different views from it. They state that the visualization of scenarios, i. e., a specific path through a set of business process models in the collection, is one of the key requirements of business process repository users.

Rulle and Siegeris [126] propose a state centric business process architecture based on our BPA approach. They use trigger relations to depict interrelations between processes that initialize a change of state of data objects. For their analysis they adapt our BPA correctness criteria in Section 4.7 with rules for their state centric PA and employ our BPA-patterns and anti-patterns of Section 6.1. The aim of their approach is to find a model that assures correct state transitions in business process interactions, similar to our data BPA approach presented in Chapter 8.

Lind and Goldkuhl [76] propose a framework for describing business process interaction. Their framework consists of interdependent layers and is derived from a language/action-oriented approach for business modeling. Lind and Goldkuhl describe business transactions and exchange between different actors based on a speech act which defines a request, the fulfillment, and confirmation of a transaction. The interaction between business processes is described with generic patterns that are structured in five layers aiming at highlighting the essentials of businesses.

NAVIGATION AND VISUALIZATION. Research in [96, 142, 60, 104] deals with navigation in process model collections and visualization of PAs.

Melcher and Seese [96] structure process model collections by clustering process models along their process metrics. They use a hierarchical clustering algorithm and visualize their clusters in a heatmap. This provides navigation structures for the process model collection. In a similar way, Srivastava and Mukherjee [142] cluster process models and their documentation for organizing the information found in the process models. This method results in process model repository structure in which similar process models are found in the same cluster, i. e., process category. It is a multi-faceted classification approach (see page 44).

Hipp et al. [60] propose a concept for navigating in process model collections with a zoom functionality and filters similar to Google Earth for geographic maps. Their navigation concept is based on four hierarchical levels, being the process world on top level, the process area on second level, the process itself on third level, and the process step on the fourth level. It allows zooming in and out of a process, e. g., from process model to process map when zooming out. Hipp et al. do not elaborate on the structure of the process model collection and the underlying model that connects the different layers of the process model collection. Their approach requires a clear structure, e. g., defined in a PA.

In his master thesis, Milde [104] elaborates on the requirements for visualizing PAs, i. e., the structure of a process model repository. He identified five use cases for visualizing the structure of a process model repository: visualizing and understanding process interdependencies, PA quality analysis, comparing new and old PAs, navigating in the process model collection, and business process re-/design. Depending on the underlying PA methodology, visualizing business process architectures facilitates the understanding and restructuring of process model repositories as information on the impact of changes can be gathered. However, only very few process modeling tools allow the visualization of PAs.

Providing an overview, summarizing and visualizing process information are highly relevant requirements for PAs. According to Baeza-Yates and Ribeiro-Neto [5] providing browsing capabilities leverages information about the information collection by putting information into context with its environment. In contrast to Baeza-Yates and Ribeiro-Neto statement, most PA approaches provide single model classification but ignore contextual information from process interdependencies.

BUSINESS PROCESS MODELING TOOL CAPABILITIES. Luebbe and Schnaegelberger [81] present an overview of 22 business process modeling software tools. They examine the functional and non-functional features and capabilities of business process modeling tools as well as their cost factors. The examination is based on a functional and non-functional requirements consisting of about 200 criteria of the consulting company BPM&O. From the information given about each tool and the features announced, most of the tools offer only one dimensional classifications in folder structures and allow for creating process landscapes or value chains only manually. The survey gives an insight on the capabilities of modeling tools in practice.

3.3 CHOREOGRAPHY APPROACHES ON MODEL LEVEL

In surveys and PA approaches presented above, the need for depicting relationships and interdependencies for clearer specification, or PA analysis was stated. In this section, we examine the capabilities of process and service modeling approaches on process model level to identify further requirements that should be propagated to the PA level.

In the last decade, the focus of BPM research on process model level extended from modeling, analyzing, verifying, or aligning single business processes to the interaction between two or several processes or services. Several approaches were developed that focus on modeling and describing the behavior of interacting services and processes [155, 10, 9, 144], processs choreographies [26, 24, 28, 23], proclets [152, 151], approaches with execution focus like BPEL for Choreographies (BPEL4Chor) [29] and formal approaches that also allow for the verification of structural and behavioral properties like workflow modules [91, 158], interaction Petri nets [27, 28] and Open nets [94, 95, 92, 164, 78].

P2P APPROACH. In one of the early works on inter-organizational collaboration, Aalst and Weske [150] highlight the need for coordinating inter-organizational workflows. They provide a top-down approach in which all collaborating partners agree on a public workflow, specified in a workflow net that describes their interaction and only the tasks that are of interest to all parties. According to the domains involved in the public workflow, the workflow is grouped and the different public parts are related to each other resulting in an inter-organizational workflow net. In the third step the public part of each workflow is decomposed into a private part under the condition that it is a sub-class of the public workflow under projection inheritance. By following these steps all private parts that are created are correct by design and it is assured that the overall interaction is correct as well.

SERVICE INTERACTION PATTERNS. The research from Barros et al. [10] and van der Aalst et al. [155] introduces the foundational concepts of service interaction. They describe commonly found service interaction patterns and examine service properties mainly in regard to inter-organizational service composition and service compatibility. Among their 13 basic service interaction patterns Barros et al. [10] introduce also three multi-transmission patterns and three one-to-many receive or send patterns. Aalst et al. [155] extend the collection of service interaction patterns and propose 23 service interaction patterns including anti-patterns. They also look at multi-instance correlation patterns in regard to one-to-one service correspondences and mainly deal with service refinement, replacement, and integration. Besides service patterns, Aalst et al. [155] provide means for verification of inter-organizational service interaction compatibility (deadlock freedom) and single services for controllability. Controllability is the ability of a service to be composed with another service without running into a deadlock. Both approaches look at the inter-organizational service composition between two processes in regard to their message exchange.

WORKFLOW MODULES. Workflow modules are used in research for modeling inter-organizational business process and service interaction. The distribution of work in inter-organizational processes, in this case workflow modules, poses the question of initialization, termination, and compatibility.

Glabbeek and Storck [158] analyze workflow module interaction in regard to proper global termination and examine the invocation relations between processes. To model the request-reply-interdependency between process and sub-process they extend the workflow module formalism with IO ports and query ports. A query port consists of two transitions and two places that portray the state of the query. Transition connect to the IO port of the according sub process and invoke their sub-routine. After having processed the request the sub-process sends back its result to the waiting process through its output place that is part of its IO port. To differentiate between cases that are handled by the cross organizational workflow net they introduce case IDs and guards that ensure that transitions only fire with correlating tokens. By determining whether all participating workflow modules can locally terminate, they verify if the combined large cross organizational workflow net has terminated.

Based on the defined properties, Glabeek and Storck [158] introduce query nets as properly terminating workflow modules. If all workflow modules in an process interaction are query nets, then the interaction of all workflow modules can properly terminate [158].

Using workflow modules Martens [90, 91] investigates the composition and replacement of cross organizational web services and their interaction in regard to their usability, compatibility, and equivalence. The general idea is to compose workflow modules such that the composition results into a workflow net that is at least weak sound.

To replace one web service with another, their behavior needs to be similar, which is expressed by the notion of simulation/equivalence. Martens [91] introduces communication and usability graphs to evaluate the behavior of the composite workflow modules. A workflow module's communication graph describes its external behavior and its usability graph describes its modules usable behavior. The equivalence of two workflow modules' behavior is verified by a simulation relation between their communication graphs.

Workflow modules support the modeling and analysis of two or more interacting web services. In [158] the need for modeling invocation relations is highlighted whereas Martens [90, 91] focuses at the message exchange between services. To examine the quality of interaction between web services and processes according properties were defined. The intention to realize particular processes by the composition of different web services and the ability to replace one web service with another are the main drivers for using workflow modules which allow describing and examining such settings. This method focuses on one-to-one message exchanges. It does not directly support one-to-many communication, multi-instance communication, or broadcasting information to several interaction partners.

INTERACTION PETRI NETS. Interaction Petri nets are an extension of Petri nets. They were developed to provide a formal model for describing and verifying global interaction models between several cooperating business partners [27, 24, 28]. Interaction Petri nets are labeled Petri nets that represent process choreographies, i. e., the sequences of message exchanges between two or more actors. The actors (sender and receiver role) involved in the message exchange and the message type are referenced by the label of the transition. A message exchange is represented by the firing of a transition, i. e., the sending and receiving of a message is one atomic step [24, 28].

Based on this formalism Decker and Weske [27] define the properties local enforceability and realizability for interaction models. Local enforceability for interaction models assures that the initiator of a message exchange must have participated in the previous message exchange as sender or receiver. Realizability describes the possibility of representing a choreography through the sequence of behavior models under specific constraints, i. e., message exchanges performed by defined roles. A choreography that is realizable is always locally enforceable, i. e., the set of realizable choreographies is a subset of choreograhies that are locally enforceable [27, 24, 28].

Interaction Petri nets provide a powerful formalism to examine inter-organizational interaction [27, 24, 28]. The approach takes an actor based view and considers one-to-one interaction between cooperating entities. As one of the few approaches it also provides means to reference message types but does not consider data aspects any further.

OPEN NETS: OPERATING GUIDELINES. A large body of work on service composition is based on the Open net (ON) approach [93, 79, 155, 92]. ONs are used to describe and analyze inter-organizational service composition and inter-organizational process behavior. For instance, Baldan et al. [7] define ONs to model inter-organizational process behavior and describe the composition of two interacting ONs along their common sub-nets.

Dealing with correct service composition and behavior of collaborating services Massuthe [94, 93, 92], Lohmann et al. [79], and Lohmann [78] use ONs and establish operating guidelines as a formal description of requirements for service composition. Operating guidelines were introduced to specify the behavior of the set of compatible services without the need to reveal the internal structure of a service and hide irrelevant information [92, 78, 155]. An operating guideline for a service s is a Boolean annotated service automaton that defines the properties of the set of compatible services to s. It is an efficient representation of the compatible set of services of a service s that are behaviorally compatible with s, i. e., their composition with service s is deadlock free. In this regard also the notions of strategy and controllability of services are introduced in [92]. A service s' is called a strategy for a service s if the composition of both services is compatible. If such a compatible service s' exists for a service s, then service s is controllable. Weinberg [164] presents a technique to efficiently analyze the controllability of WS-BPEL processes by transforming them to ONs and calculating their operating guidelines.

Operating guidelines specify the paths in the state space that reach the desired final marking as Boolean annotated service automaton. Massuthe uses operating guidelines for supporting the discovery and deciding the suitability of services [92]. An operating guideline can be used as information to be published in a service repository while hiding internal structure of the published service [94, 92, 78].

Lohmann [78] elaborates on service composition by providing techniques for supporting the design phase of service composition and automatically verifying correctness of service compositions as well as completing partially specified compositions and fixing incorrect compositions. Additionally he introduces a technique to deduce correct local service descriptions from globally defined choreographies [78]. He employs operating guidelines to characterize the set of all compatible partners of a service and in his correction algorithms.

Aalst et al. [155] use ONs to describe and analyze the previously mentioned service interacting patterns. They establish a formalization of strategy, controllability, and accordance for exposing services, to refine and replace, as well as to create adapters. Accordance allows a service s to be replaced by another service s' if the replacement service s' is compatible with all services that service s is compatible with.

PROCLETS. Proclets are lightweight processes, process fragments, that were developed to support the modeling and implementation of complex processes. By dividing complex processes into smaller simpler processes, proclets, the complexity of the processes is reduced. In the same time stronger emphasis is laid on the modeling of interaction between those lightweight processes [89]. The focus on the interaction between proclets allows for more flexible process combinations, such that many cases do not need to be squeezed into one large process definition which is too restrictive according to Aalst et al. [151, 152] and Mans et al. [89]. This is especially helpful in the environment of hospitals where cases have a high variability in regard to the diagnoses [151, 152, 154, 89]. The most recent research on proclets deals with specifying the interaction between process on process instance level and the implementation in Workflow Management Systems [154, 89, 87]. Especially the interaction between proclet instances and their relationships are in the focus of research.

The communication between proclets is defined on instance level. Proclets allow modeling the sending of a message from one sending instance to many receiver instances [154, 89, 87]. The representation of a one-to-many relationship in regard to different partner proclets is specified by defining several one-to-one message exchanges. This work emphasizes the need to focus on the interaction between internal processes. Although their approach rather focuses on dividing single complex processes into smaller ones and recombining those proclet instances in a flexible way, the requirement of depicting the interaction between processes, one-to-many, and many-to-many interaction, as well as looking at resources and data should be transferred to large process model collections and their inherent process interdependencies. The proclet approach can be situated on the process model and implementation level due to its very detailed description of process steps and its detailed specification of content and instance interaction. Recent research by Aalst et al. [154] and Mans et al. [89, 87, 88] include implementations of the proclet approach for Workflow Management Systems.

BPEL4CHOR. Decker et al. [29] introduce BPEL4Chor, an extension of BPEL, to represent business process choreographies in BPEL. They also provide a transformation of BPMN to BPEL4Chor to join the visual representation of BPMN with the technical expressiveness of BPEL. This approach focuses on the implementation of the interaction between cooperating organizations.

BPMN CHOREOGRAPHY DIAGRAMS. In Section 2.3.2 BPMN choreography diagrams were introduced. They were derived from work on process choreographies and interaction Petri nets [27, 24, 28]. Choreography diagrams are used to model the interaction between cooperating organizations. They provide an actor view and depict the message exchange between the different partners [165]. It allows to model broadcasting messages from one sender to several partners, however, the receiving of several messages from different partners cannot be modeled with one choreography task. The private processes that realize the message exchanges in each company are hidden in this diagram type. In this regard, one cannot determine how many processes are internally involved in a choreography.

DATA IN PROCESS CHOREOGRAPHIES. So far we have looked at the choreography approaches that focus on the control flow of interacting processes and composed services, or proclets. In recent years new approaches that also look at data aspects in choreographies and process interaction have been developed. So far the work on data dealt with consistency between data model and control flow of single processes [48, 77] or the synchronization of several object lifecycles (OLCs) of different data objects in one single process [103]. The primary aim of most approaches is to generate models with consistent and correct data and control flow perspectives. Research on data aspects in process models can be grouped along two main streams, activity centric and object centric process modeling. For instance, Nigam and Caswell [111] and Cohn and Hull [20] introduced business artifacts, an object-centric approach, that Yongchareon et al. [173] refined by providing a formal framework for process specification utilizing synchronized OLCs. Each involved data object with a corresponding OLC correlates to one process and transitions of different OLCs required to be executed together define the synchronization. This is done for business artifacts by defining synchronization edges [173].

PHILharmonic Flows [73] and COREPRO (COnfiguration based RElease PROcesses) [108] are object-centric process modeling approaches. PHILharmonic Flows [73] focuses on describing processes with OLCs. The interdependencies between different data objects are presented in terms of a hierarchical data model. COREPRO [108] assumes one high level system process model that depicts hierarchical relations. Processes and their sub-processes are represented as OLCs. To maintain them and provide consistency among the hierarchical levels of process and sub-processes and their data objects, interdependencies are defined between the OLCs of the data objects. The approach provides support for the maintenance of data objects, flexibility, and re-use on process model and instance level. Both methodologies, PHILharmonic Flows and COREPRO, use hierarchical structures to define the interdependencies between the data objects and show the necessity of hierarchical relations for a better and easier maintenance.

Considering the data perspective similarly to object-centric approaches, activity centric models can be enhanced with data objects and their states. Approaches by Eshuis and van Gorp [48] and Liu et al. [77] propose extracting OLCs from activity-centric process models for each used data object and assess them for consistency. Meyer and Weske [102] introduce the data correctness notion of weak conformance and develop a technique to analyze if an activity in a process model is able to access the required data objects in the expected state. They also provide a verification algorithm that combines the checking of data conformance and control flow soundness.

These research endeavors on data aspects show the importance of data in business processes. Presented examples of object-centric and activity centric approaches focus only on single process models. However, the exchange of messages between several processes is based on data objects and information exchanged, so that data interdependencies exist between processes. Data interdependencies and the exchange of data between processes become more important but have not yet gained major research attention.

Knuplesch et al. [69, 68] introduced a data layer into process choreography diagrams. They propose data aware choreographies in which virtual data objects define the message exchange between processes. The information specified in the data objects is used to determine the routing of the control flow. For the analysis of the object flow they propose Data-Aware Interaction Nets (DAI) and provide a transformation between Data-Aware choreographies and DAIs. This allows them to analyze choreographies for realizability and termination considering data aspects during message exchanges [69, 68].

Bridging the gap between activity-centric and object-centric process approaches Fahland et al. [51] use proclets to model artifacts. They provide a technique to check conformance by decomposing artifact-centric processes and mapping them to an interaction conformance problem. This approach is based on execution logs and aims at conformance checking between process execution level and model level. Meyer et al. [103] provide means to model and enact complex data dependencies in activity-centric process models by providing a global data model known from object-centric approaches.

Data interdependencies depict important information, e. g., on the status of an order or the production of a good, and need to be reflected in business process architectures. Consistency, conformance, and correctness between data and control flow perspective should also be examined in regard to process interactions for avoiding erroneous behavior. In Chapter 8, we introduce a technique to extract data interdependencies between business processes and depict them on PA level.

3.4 BUSINESS PROCESS MODEL ABSTRACTION

Business process model abstraction is a technique to provide an overview of the most important aspects of a process model and reduce the complexity by either eliminating unimportant aspects or aggregating aspects of interest, i. e., the generalization of detailed process models [136]. Business process abstraction is a bottom-up approach of providing simplified process models.

PAs could be considered a high level abstraction of process models as many PA approaches group and aggregate process models by their business objects or functions for example. Several process models are then represented by the function or the business objects they share.

Within the scope of the process model level, different algorithms have been proposed and refined for abstracting process models in [135, 115, 116, 139, 137, 138, 140, 141]. Smirnov et al. [138] highlight the need for a fast overview of a process's main characteristics based on an empirical survey of health insurance workers and validated by BPM consultants. Business process architectures have similar aims, e. g., offering information and an overview of the main characteristics of processes in a particular category. In [141] Smirnov et al. elaborate on the use of process model abstraction and present a classification of process model abstraction use cases. Similar to PAs the aim of abstraction is to highlight important aspects of a process model, hiding the unimportant parts, and hence reducing complexity.

Polyvyanyy et al. [116, 115, 117] improve the technical algorithms of Vanhatalo et al. [159, 160] for process model abstraction using either Process Structure Trees (PST) or Refined Process Structure Trees

(RPST), techniques for decomposing process models into smaller fragments that can in a later step be consolidated and abstracted.

A large part of research on business process model abstraction was developed by Smirnov et al. [135, 139, 137, 138, 140]. They present different formal techniques for process model abstraction that allow firstly decomposing complex process models into fine grained components and secondly abstracting those components into a generalized abstracted view of a process model in regard to structural, behavioral, and semantic aspects [139, 138, 140, 137].

In a similar context, Reijers et al. [122] discuss modularization of process models in regard to better comprehension of process models. They conducted an experiment to examine if modularization of process models, i. e., using sub-processes, fosters understanding of process model. The results of the experiment indicate that modularization has a positive effect on process model comprehension by hiding information in sub-processes. This was expecially apparent when sub-processes were used extensively. Based on their findings, they investigated measures to identify process model fragments that are suitable for being captured in a sub-process. For this, the process model metric *connectedness* appeared to be the most promising. In this context connectedness of a sub-process describes that the nodes of the according process model fragment (sub-process) are stronger connected with each other (e. g., by number of arcs) than to the nodes outside of this fragment (the other parts of the process model).

In a different approach, Reijers et al. [123] present a one-to-many process model aggregation algorithm for easier maintainability and management of process models in a process repository They introduce aggregated EPCs (aEPC), an extension of EPCs, that describe a set of process models by their product hierarchy, along generalization/specialization relationships between node and leaves in the product tree. By visualizing context information of the original models and deriving matching product hierarchy trees they improve end user comprehensibility of the aggregated models.

Eshuis et al. [47] and Reichert et al. [121] present two methodologies to provide customizable process views on large business processes. Eshuis et al.'s [47] scope are inter-organizational process collaborations for which a company needs to publish their public process that hides secret and unrelated business process information, similar to the private/public approach mentioned earlier [150]. Their approach consists of two steps; first creating a non-customized view on the internal process by aggregating internal activities, and second customizing the resulting process by omitting and highlighting activities that are not requested by the collaboration partner [47]. Deriving the customized view requires input from the consumer who decides which activities are important to him. The process is then tailored to the consumer's needs by omitting the irrelevant activities. Eshuis et al. [47] provide a formal definition of the aggregation and customization algorithms.

Reichert et al. [121] present the Proviado view mechanism to create personalized views on large and complex business processes by using graph reduction and aggregation techniques. Similar to [47, 135], they hide or aggregate information in a respective business process model to emphasize the required information for the user. In the forefront of the framework is the visualization of complex process models down to the implementation on instance level to achieve personalized views on a business process based on parameterizable operations, e. g., that shows a user only the activities the user has to perform. Their parameterizable operations allow users to configure their view and the properties that the personalized view should comply with. For instance, for the personalized view they define three different node dependency properties: dependency erasing, generating, and preserving. Reduction operations are always dependency erasing whereas aggregation operations can be of any of the three types [121]. Both approaches use abstraction algorithms to provide customized business process views. Their scope lies on single business processes and do not take business process interdependencies into account.

3.5 EVALUATION OF EXISTING APPROACHES

In this section, we discuss and evaluate the presented approaches. We highlight aims, requirements, and gaps, especially from the initially presented surveys [53, 54, 56, 83, 85, 84, 35].

The main aspects identified in the PA surveys and PA evaluation frameworks are: aim of methodology, framework characteristics, scope of the approach, type of relationship and interdependencies depicted in the framework, and the elements involved. Especially horizontal interdependencies, consistency between layers, visualization and overviews as well as formal specification of relationships were mentioned as gaps in PA research and are also reflected in the aspects that will be used for classifying related work. Table 7 in Appendix A summarizes related work and list the according characteristics of each approach.

- *Aim of methodology*: visualization, definition of relationships, overview, analysis, classification [56, 70, 83, 84, 35]
- *Framework characteristics*: ease of maintenance, consistency, comprehensibility, and clear specification of relationships (in-/formal) [56, 54, 70]
- *Scope of methodology and application*: single models or several models, applied bottom-up or top-down, intra-or inter-organizational

- *Type of relationship between elements*: hierarchical or horizontal, e. g., sequence order, activation, message flow, input and output [56, 70, 83, 84, 35]
- *Relationship elements*: the elements that take part in the relation, e. g., activities, objects, process, input, output, etc. [56, 70, 83, 84, 35]

The *aim of methodology* describes the main use cases that are supported by the research approach, e. g., classifying processes and providing an overview on a process model repository. The *framework characteristics* describes the framework in more detail, e. g., if there is a formal specification of relationships and elements, if it provides support for maintenance, or if its describes mechanism to assure consistency between different layers of a PA.

Scope tells us if the approach focuses on only single process models or on several process models at once and if it is intra- or inter-organizational. Approaches that focus only on single processes do not take interdependencies between processes into account whereas approaches that consider several processes usually depict a relationship and interdependency between processes. For instance, a PA approach that classifies process models each one by one into categories is considered to have a single process model scope as the classification process always looks at only one process at a time.

The category *type of relationship between elements* describes relationship types that the approach under examination specifies, e. g., in which way processes are connected to their category group(s) (hierarchical) or other processes (horizontal).

Relationship elements describe the elements involved for defining the relationships. A function based classification relates processes to business functions so that the element of focus would be the business function that may group several processes into one category. The process is in a part-of relation to the business function for example.

From the presented research approaches we derive four groups:

1. Hierarchical PA classification approaches considering single processes
2. Single process model approaches for visualization, navigation, and configuration
3. Horizontal PA approaches considering multiple processes
4. Detailed process model choreography (multi-process) approaches with formalism

3.5.1 *Hierarchical PA Classification Approaches Considering Single Processes*

Nearly all the EA and PA approaches presented are based on a classification of single business processes along different aspects (business object, action, function, reference, goals). Most of the approaches employ a functional classification of the process models but also business objects are often used for classification purposes as has been also found by Dijkman et al. [35]. A third of the PA approaches use reference models to classify their approaches which were also functional based in many cases. Although regarded as very popular and easy to use by practitioners, the guidelines of reference model based approaches, were rather considered not useful according to [35] . Our observations of related work coincide with Dijkman et al.'s findings.

Less than a quarter of approaches classify their business processes according to business objects, e. g., the product produced or important data objects. Only few classify the processes according to goals or actors like departments or specific roles. The majority of these approaches provide some sort of informal hierarchical aggregation and decomposition as relationship type between the process model and its category, e. g., specialization, generalization, aggregation or decomposition.

Srivastava and Mukherjee [142] and Melcher and Seese [96] employ clustering algorithms to provide an automated classification of the process models for easier navigation and visualization. Automated extraction of such navigation systems is an advantage on the one hand as it saves time and effort; as a result the maintenance of the process collection may be simplified. On the other hand the resulting clusters in most cases change when the input set of the clustering algorithm changes, i. e., when process models are added or removed from the process model collection. This would result into constantly changing navigation structures which would confuse the users of the process model repository. These two clustering approaches are the sole classification approaches with a formal specification in form of a similarity function.

The common aim of the approaches in this group is to provide a means to structure and manage process models within process model repositories and support the re-/design phase of process modeling projects. In many cases the classification of the process model is also used for navigating in the process model repository and to provide an overview of the existing process models.

3.5.2 *Single Process Model Level Approaches for Visualization, Navigation, and Configuration*

This group encompasses process model visualization, process model (repository) navigation and all business process model abstraction approaches that focus on single process models. Their common aim is to reduce complexity and enhance comprehensibility in regard to process model understanding (filtering unnecessary information) by providing aggregated, customized, or filtered views on a process model. This is similar to the aims of most PA approaches but situated on the lower process model level. All of the visualization approaches are bottom-up as they build on existing process models and their metadata. The process navigation work by Hipp et al. [60] describes a navigation architecture with different views. The research on business process model abstraction [116, 115, 117, 135, 139, 137, 138, 140], as well as business process customization and configuration [121, 47, 8] is clearly specified in formalisms. Most visualization techniques build on the underlying specified classification scheme.

3.5.3 *Horizontal PA Approaches Considering Multiple Processes*

All PA approaches that consider several processes and describe horizontal interdependencies between processes or their categories fall into this category. Less than a quarter of PA approaches examined consider horizontal interdependencies between the elements of interest. The lack of approaches depicting interdependencies between processes was also identified by some of the research papers as main gaps in PA research which coincides with our findings here.

The horizontal interdependencies described by these approaches are manifold. They focus on relations between actors, data objects, or processes and their activities. The survey by Maddern et al. [83] is also grouped under this category but does not describe any concrete PA approach. On PA level Lind and Goldkuhl [76] present one of the few approaches that describes business transaction and exchange between different actors based on a speech act which defines the request, the fulfillment, and confirmation of a transaction. Jacobs et al. [63] define horizontal relationships between processes of different reference process models in order to create and populate an enterprise repository.

Rulle and Siegeris [126] use our BPA pattern and anti-pattern analysis technique for assuring correct interdependencies in their configurable PA. The RIVA approach by Green and Ould [55, 57] provides similar horizontal relationships like activates and interacts with between processes.

Apart from Rulle and Siegeris [126], Jacobs et al. [63], and Green and Ould [55], only Moreira et al. [105], Schmelzer and Sesselmann [130], and Scheer [127] name horizontal relations but do not specify them and their application in detail. Therefore those approaches were considered having a single model focus. Except from Rulle and Siegeris [126], all other multiple model scope approaches are informal. Three of them focus on the analysis of the interdependencies described.

Except from Milde [104], none of the examined visualization approaches considers process interdependencies and their visualization.

3.5.4 *Detailed Process model Choreography (Multi-Process) Approaches with Formalism*

A large group of examined approaches provide means to portray horizontal process interdependencies between processes or actors. These approaches provide strong formalisms for the composition and analysis of inter-organizational business process or service interaction. The approaches in this group are situated on process model level and provide a detailed description of external flows. In most cases the relations defined are one-to-one message exchanges and composition of two services or processes. The works by Decker et al. [24, 28] also allow for the specification of one-to-many message exchanges between several actors. Here the one-to-many exchange refers to the sending of one message to several receivers.

The Proclet approach developed for specifying the execution of complex process models, also allows for modeling the interaction between process fragments in such a way that one proclet instance can communicate with many instances of another proclet. Here the one-to-many relation means the sending of one message to several instances of the partner process [152, 88, 89]. The sending of one message to several partners can only be modeled by adding a relation for each pair of sender and receiver and assigning the same message object to the channel.

The choreography approaches examined mainly focus on smaller scenarios. The need for modeling choreographies originates from the requirement of specifying the interaction with an external customer or service provider. This reduces the scope of these approaches to defining a single process from each participant that fulfills the behavioral contract specified in the choreography. It is assumed that the internal processes work correctly and that the partner process is ready to receive the information sent. A choreography ends with the fulfillment of the agreed interaction contract and it is not of interest which processes are triggered internally in an organization.

In this group we find also choreography approaches that specify and consider data objects in the process interaction. A fairly new research area is the integration of data aspects in choreography modeling which shows the increasing importance of data aspects and resource modeling while looking at process interdependencies.

Most of the approaches in this group define message exchanges between processes, services, or actors. The explicit modeling of activation or triggering is not considered. Only proclets allow indirectly modeling the activation of a process fragment.

The business process model abstraction methods and the configurable process views lay a strong focus on reducing complexity of the process models by eliminating irrelevant information and aggregating relevant aspects of interest. The overview and visualization of parts of the process model are especially important.

3.5.5 *Observed Gaps*

In the past ten years, we could observe a strong focus of research on the examination of single process models. The field of PA research deals with structuring, managing, and visualizing many process models in an overview within a process model repository. The majority of approaches provide a classification of the process models along different aspects, e. g., business object, function, department, or goal. Rarely a holistic view has been taken as each process model is classified individually. While choreography approaches are very detailed and cover the inter-organizational interaction, the internal interdependencies between processes have not been examined in detail. For managers, IT-, and business experts, it is valuable to get an overview on how a case is routed through their organization and which processes interact with each other to realize a public service or the production for a good. Similar to business choreographies that look at external inter-organizational interaction, internal process interdependencies and their interactions have to be identified, visualized and examined. PA approaches that reduce the complexity and detail of process models, but still depict the context in which they operate, i. e., their interdependencies in terms of input/output, message exchange, or resources for example have not been proposed yet.

We could deduce from the presented business process choreography approaches the importance of depicting the composition and analyzing the behavior of the composed processes. Some of the choreography approaches highlighted the need to depict multi-communication, i. e., broadcasting and multi-casting of messages to several instances of a partner process or several partner processes [89]. In process model repositories information on interdependencies and multi-communication between business processes is available to an extent that goes beyond the scope of choreography approaches. So far, this information is not leveraged on the PA level.

Often the detailed process model layer and the abstract PA layer are not connected and modeled independently from each other. Common PAs do not have a formal foundation or a clear specification to depict the hierarchical and horizontal relations. So far, hierarchical and loose horizontal relations have been captured in an informal way. The strength of an underlying formalism is obvious from the formal approaches on service composition and design, and choreographies [94, 79, 95, 155, 28]. Most of the choreography approaches are based on a strong formalism and a range of correctness criteria has been presented as well as techniques for verification. Similar correctness properties as we find on process model level do not exist for PAs. In rare cases, manual analysis was proposed as in [70, 55, 126]. This is due to the lack of formal foundations, correctness and consistency criteria, as well as clear specification of relationships between elements of a PA. On PA level analysis often refers to the evaluation of assigned key performance indicators.

Except from the concrete guidelines given by Dijkman et al. [35] and zur Muehlen et al. [106], most of the PA approaches describe only their informal methodology on the relations and structure of the process model collection but do not specify a concrete technique how to get from process models to the PA level and vice versa. Consistency criteria between PA and process model level do not exist and means or guidelines for ensuring consistency between the layers are not found in PA literature.

Many PA approaches create a break in their decomposition from business functions to process models and their process activities. The starting point are business functions that are then decomposed until they reach the granularity of process activities, and cannot further be decomposed. Such a decomposition, however, contradicts the aim of companies of becoming more process-oriented and being structured along processes. The process oriented focus of the lower process model repository levels and the inherent rich information is not propagated to the PA level of process model collections where we mostly find classification along different aspects.

The process model abstraction approaches show that it is possible to reduce complexity while preserving aspects of interest and hiding, aggregating, or eliminating unnecessary information. PA approaches hide process information in their classification scheme in such a way that the abstracted information, i. e., the representation of processes on PA level, is so abstract that all information is lost and not useful except for navigation purposes. Some interdependency concepts are realized by the RIVA methodology by Green and Ould that includes a mix of actor and process view on high level but it does not define the influence of such relations on processes and actors. Business process model abstraction approaches focus only on the abstraction of single processes. A PA approach that provides an abstract representation of processes and preserves and clearly defines their horizontal interdependencies does not exist.

One-size-fits-all approaches like business process landscapes, value chains, the categorization into departments or the classification into main, supporting, and management processes, do not highlight any company specific aspects or important company characteristics due to their generality. Neither process characteristics are found on PA level nor any interdependencies that could be extracted from the process models.

Easy maintenance and comprehensibility are reported as important factors when creating PAs but how this can be achieved has rarely been discussed in proposed PA approaches. Automated creation or extraction of PAs has not been proposed yet, except from the clustering approaches by [96, 142] and the process to enterprise architecture alignment approach by [63].

Valuable and far reaching approaches have been presented in PA research. Intrinsic classification techniques for organizing, structuring and managing large process collection have been proposed. However, PA research in comparison to existing research and approaches on process model level, is still in its infancy.

In summary, the research gaps encompass the weak preservation of processes and their characteristics on PA level, the informal specification of horizontal and hierarchical relations, rather single process model focused approaches than holistic approaches, non-existent quality criteria for PAs and analysis techniques, weak linking of PA and process model level without clear consistency criteria, and the lack of automation and support for maintenance and analysis.

3.6 SUMMARY

In this chapter we introduced a broad range of valuable research on PAs, enterprise architectures, business process model abstraction and configuration, and business process choreographies. These topics that all partly touch the field of PAs. We provided some background on existing techniques and approaches on PA and process model level. We identified different requirements for PAs and gaps in PA research. Some of which will be the foundation for the PA approach that we will present in the following chapters.

Part II

CONCEPTUAL DESIGN

4 BUSINESS PROCESS ARCHITECTURE

This chapter is based on the published papers [40, 38, 45, 43]. In this chapter, we present the core concepts of our BPA approach. After an description of our motivation, requirements, and assumptions, we introduce the main elements and relationships of our BPA approach and the rational behind it. Together with the next chapter it provides the foundation of our BPA approach.

4.1 MOTIVATION

BPM aims at improving an organization's operations and processes, increasing efficiency and reducing costs [114, 165]. With the advent of BPM in the private and public sector, large process model collections have been established in many companies and public organizations. We have noted that already many hierarchical PA approaches exist which focus on the classification of single process models and provide a structure for process model repositories. However, looking at all the process models one by one in large process model collections does not reveal all their valuable information. We identified that only few PA approaches consider horizontal interdependencies between processes. So far, the visualization and analysis of internal process interdependencies throughout the whole organization (end-to-end) has not been examined in detail. Business process owners are confronted with questions like:

- Which processes influence the execution of my business processes and how can I streamline them?
- How does the re-design/improvement of my process affect other related business processes?
- While I know my individual processes to be sound, are my processes also sound in their relations with each other?
- How does the customer journey through the administration offices look like as a whole, when, for example, that customer wants to open a new enterprise?
- Where should one business process start and another begin and how does this affect assignment of responsibility?
- What are required resources or expected costs for the execution of a group of interrelated business process?

- What is the risk of failure of my business process in regard to its interdependent processes?

These questions cannot be answered with traditional PA approaches. We take a different view with our novel BPA approach by studying the interrelationships between the business process models of an organization. Only by taking a holistic view on a process model collection, and its inherent process models, our BPA approach shows and highlights inter-process dependencies, and hence extends and leverages the stored information. This maximizes the value of the effort put into the business process elicitation process. Taking a holistic view is especially important for the optimization, analysis, or re-design of a company's process landscape. Such an approach has a larger impact than single process optimization efforts as the interaction between processes is harmonized avoiding a negative performance impact on each other. In the following, we elaborate on the requirements and introduce the foundations of our novel BPA approach. As noted before, we refer to our approach as Business Process Architecture (capitalized) and abbreviated as BPA whereas we refer to common business process architectures and the general research field as business process architecture (lower case) and abbreviated as PA for better reading and differentiation.

4.2 REQUIREMENTS FOR BUSINESS PROCESS ARCHITECTURES

Based on our evaluation of related work, we introduce the requirements for our BPA approach. We envision three user types to use our BPA approach; managers/business process owners, business process experts, and the business process users/employees. Managers and business process owners need a high level view on their business processes. For them it is important to easily grasp information on their processes and use this information to support their strategic planning, calculate costs or resources, and improve the operations of the organization. They are responsible for their business processes, take a managerial perspective on the business process lifecycle and ensure that their business processes are up to date. For this they may interact with other process owners to harmonize and discuss the interaction of their processes on high level. Business process owners and managers are not necessarily highly knowledgeable in BPM and may only be BPM novices.

The main user of our BPA approach is the business process expert that leads the BPM initiative. Business process experts are responsible for the overall methodology applied in the business process elicitation project and design process modeling guidelines. Usually, the business expert reports to the business process owner. They design, analyze, and improve business processes. The third user is the common employee who uses the process model collection as job instruction and information source.

Figure 14 shows a UML use case diagram that outlines the use cases our BPA approach shall cover and the users involved. We identified four use cases from PA research. The use cases *structure and manage process models* which includes the *use of a design methodology*, and *get overview* over the process models describe the general requirements to a PA approach. The use case analysis in general is only covered by very few approaches like Green and Ould [55, 56, 57] or Rulle and Siegeris [126]. The use case *analyze interdependent processes* extend the analysis to a larger context than only single process models. The four main use cases are extended by a range of refined use cases that our BPA approach shall cover.

The use case *structure and manage business process models* is extended by the use cases re-/design process models, update and maintain, and navigate. The ordering system defined by a PA is used to navigate in the business process collections. The structuring and managing of process models in a process model repository requires the use of a design methodology. To provide a clear specification we regard a formal description of our BPA approach as essential. As the focus of our BPA approach lies on interdependencies, our BPA formalism is required to provide means to define those business processes and their interdependencies. Business processes can have many different type of interdependencies, e. g., message interdependencies, data interdependencies, or originating relations. The BPA methodology shall represent process interdependencies and retain the process oriented view and characteristics of business processes. The classification/ ordering system of our BPA shall base on those interdependencies. During our research we encountered many different process modeling notations. Hence, a our PA approach should be modeling notation independent.

In contrast to current PA approaches our approach shall be applicable top-down and bottom-up while in both cases the consistency between PA and process model layer has to be assured. The elicitation and maintenance of a process collection is a strenuous manual task that is time consuming and costly. For collections of hundreds or several thousand process models, e. g., SAP Reference Model, Dutch administration, or China CNR Corporation Limited [74], this is not efficient. Our BPA approach shall facilitate and support the maintenance of the process model collection along all different layers while considering consistency issues.

We have observed that hardly any PA approach allows for the analysis of business process architectures. Hence, we envision that business process architectures modeled with our BPA approach can be analyzed. This is captured by the use case *analyze interdependent processes* and its extensions in Figure 14. In this regard our BPA approach shall provide a technique to analyze a PA, its inherent business proces-

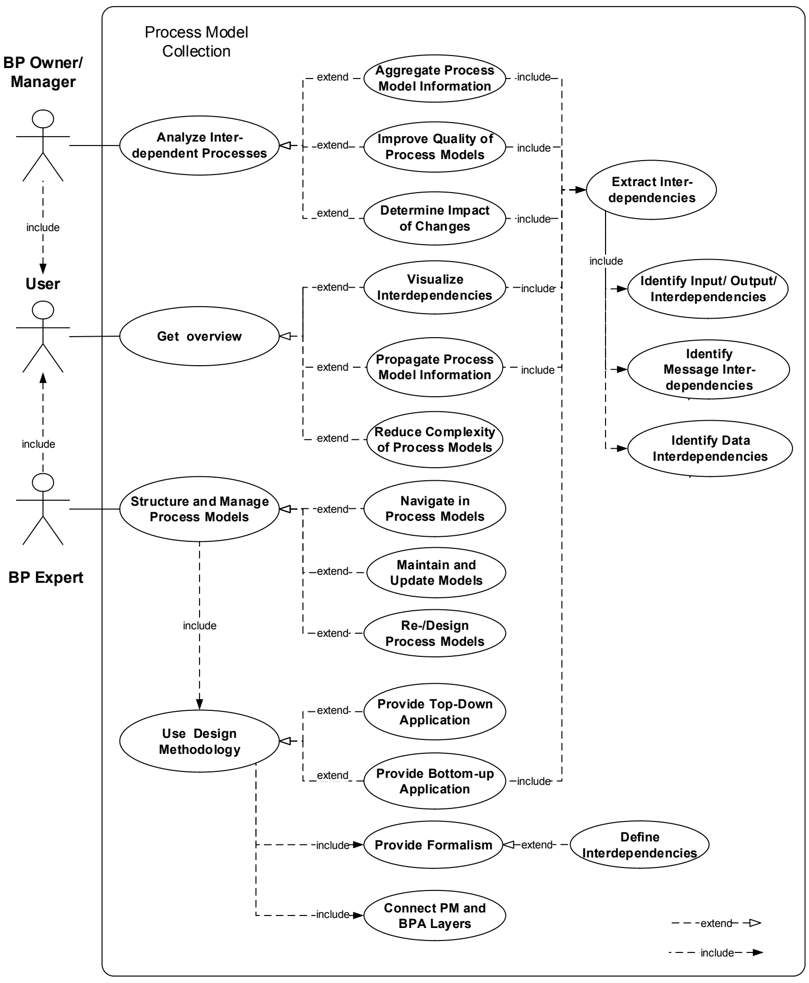

Process Model Collection
BP Owner/ Manager
User
BP Expert
include
include
Analyze Inter-dependent Processes
Get overview
Structure and Manage Process Models
Use Design Methodology
Aggregate Process Model Information
Improve Quality of Process Models
Determine Impact of Changes
Visualize Interdependencies
Propagate Process Model Information
Reduce Complexity of Process Models
Navigate in Process Models
Maintain and Update Models
Re-/Design Process Models
Provide Top-Down Application
Provide Bottom-up Application
Provide Formalism
Connect PM and BPA Layers
Define Interdependencies
Extract Inter-dependencies
Identify Input/ Output/ Interdependencies
Identify Message Inter-dependencies
Identify Data Interdependencies
extend
include

Figure 14: BPA use cases

ses and their additional information while considering their business process interdependencies. Here the analysis of the impact of change and the influence of business processes on each other are particularly important. The analysis of interconnected processes shall improve the overall quality of the business process collection as their interaction is better harmonized and adapted to each other. In this way, business processes can be streamlined considering their interdependencies.

The main idea of PAs is to provide an overview of a process model collection and present its business processes. Our BPA approach shall help to reduce the complexity of business process models and visualize the most important characteristics of process models in a simple way so that also non-BPM experts, like a new employee, are able to grasp their process universe and the information provided. The extracted characteristics and context information provide access and further entry points to a process model collection and its content.

The application area of our BPA approach in the BPM lifecycle is the design and analysis phase in which business processes are identified and elicited, as well as verified. In this phase, it shall support the structuring of a process model collection, the identification of main processes, as well as the coordination of the process interaction in regard to their behavioral and data interdependencies. The analysis functionality shall help to examine the BPA and its inherent processes and hence contribute to improve the process model quality. In a re-design phase BPAs shall help to understand the interdependencies between processes and to examine the impact of changes of the business process to other involved business processes as the performance of business process impacts or is influenced by other processes. A holistic view is here of utmost importance to maximize the re-organisation efforts and harmonize interaction between processes avoiding a negative performance impact on each other.

4.3 ASSUMPTIONS

Business process architectures define a structure and overview of a business process model collection. The structure, e. g., a classification system, can be created before modeling business processes (top-down) or it builds on an already existing process model collection (bottom-up). When following a top-down approach, our BPA approach does not imply specific assumptions on the process model collection as the PA is built before the business processes of the process model collection. In a top-down procedure, our BPA approach demands that the about to be elicited business process models meet the structural and behavioral aspects specified on PA level. For bottom-up approaches, we assume that the process model collections contain information on interdependencies that can be extracted through parsing and analyzing the business processes and their attributes. As BPAs build on this information we assume it to be available and easily accessible for all processes. Chapter 7 presents more details on our BPA methodology and elaborates on the problems and required process information for extracting business process interdependencies from process model collections.

In the following, we elaborate on our initial assumptions in regard to business process models that are stored in a process model repository. All business processes are unique and a process model collection does not contain duplicate business process. A business process may appear in several process model diagrams, however, this information can be detected. In general, we assume that process model collections consist of business process models depicting single business processes. We assume that the business processes in the process model collection are sound.

Companies that consist of several departments, and/or are situated in several locations, share the same process model repository. This means that the internal/private workflow of a business process is accessible for designing PAs. In general we assume that all processes in the process model collection can be accessed, also process models that depict operations of collaborating companies. For cases of inter-organizational collaborations we assume companies to share at least their public process, if not their private processes, for the sake of streamlining the overall interaction between the companies. This means that these processes are included in a process model collection. E. g., many companies outsource part of their operations but have access to the private business processes of their supplier company. This in a way contradicts the assumptions of choreography modeling approach from [150, 28] where only external behavior is specified and the internal processes are considered private.

Our BPA approach does not differentiate between the common structure of support, management and core processes. However, taking this classification into account, it covers all processes but mainly focuses on the core processes.

4.4 CONCEPTUAL DEFINITION

Most of the business process architecture design styles presented in Chapter 3 depict hierarchical and to a lesser extent horizontal relations between business processes. In total, we identify four types different types of relations:

- *composition* - one business process is composed of a number of other business processes, also called sub-processes
- *specialization* - one business process specializes another
- *trigger* - one business process causes another business process to instantiate and start
- *information flow* - one business process sends information or other objects to another business process

Composition and specialization are hierarchical relations between process models in which composition defines a 1:n relation and specialization a 1:1 relation between two process models. Trigger and information flow relations are horizontal relations between process models. They depict the concrete requirement of information exchange between two or more business processes.

Information Flows and triggers are the most basic interrelations between business processes. Triggers depict that one processes instantiates and starts another process. This means that the process that is triggered cannot start by itself. It is inactive before it receives the

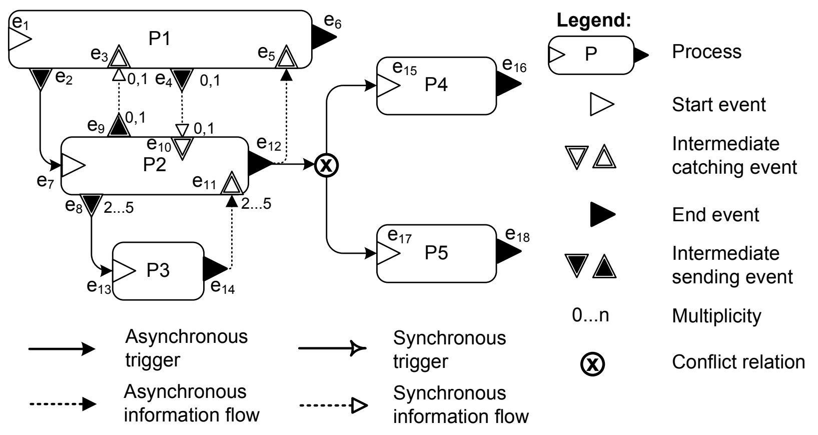


Figure 15: BPA symbols and legend

trigger that activates it. In this case it is always dependent on one preceding process that triggers it.

Information flows depict that one business process produces some information that is needed by another process. E. g., this could be a message sent from one process to another. In contrast to triggers, information flows occur only while a process is active, i. e., in the middle of a process execution. Without the expected information flow the receiving process cannot continue its execution.

With the design of our BPA approach we aim at providing a new PA concept that leverages this information from the detailed process model level to a higher, more abstract level. Our BPA approach provides a formalism to depict horizontal interdependencies between business processes that show the internal journeys through the business process of an organization regarding a specific issue like an order or the delivery of a public service like a construction permit.

For this, we need to define a formal representation of the processes and their elements that our BPA approach shall depict on PA level. In process models the sending and receiving of information objects, i. e., triggers or information flows, is depicted by sending and catching nodes. Those usually are throwing/catching events, and sending/receiving activities. Depending on the process modeling notation other types of modeling elements may be in included. In many cases, the sending and receiving of information objects, i. e., the interaction between processes, is depicted by events, that describe that something of importance just occurred, e. g., an order was received, a credit card delivered, or a sending task has been performed. Also, in many business process modeling notation the instantiation and the termination of a process are represented by events, e. g., in BPMN2.0 or EPC [112, 128]. Trigger and Information flow, as well as the involved sending and receiving nodes, can be considered as and mapped to events. Hence, in our BPA approach a process model is represented by a set of events. Each BPA process is a sequence that consists of one start and one end event, and any number of intermediate events. We differentiate between catching and throwing intermediate events. Start events and intermediate catching events are receiving events. End events and intermediate throwing events are sending events.

A relation between two BPA processes is defined by the relation between their events whereas a sending event of one process is related to a receiving event of the partner process. Depending on the receiving event, the relation between two events of two different processes is either considered a trigger relation if the receiving event is a start event, or an information flow relation, if the receiving event is an intermediate catching event. An event can be in relation with many other events of other processes. The information flow and trigger relation define a tuple of two events that consists of a sending and a receiving event. We define a BPA as follows.

Definition 29 (Business Process Architecture [40, 38, 45]) *A Business Process Architecture is a tuple* $(E, V, L, I, \chi, \mu, \pm)$*, in which:*

- E *is a set of events, partitioned in start events,* E^S*, end events* E^E*, intermediate throwing events* E^T*, and intermediate catching events* E^C
- V *is a partition of* E *representing a set of business processes*
- $v \in V$ *is a business process, consisting of a sequence of events,* $v = \langle e_1, ..., e_n \rangle$ *such that* $e_1 \in E^S$ *is a start event,* $e_n \in E^E$ *an end event, and* $e_i \in E^C \cup E^T$ *for* $1 < i < n$ *are intermediate events*
- $L \subseteq (E^T \cup E^E) \times E^C$ *is an information flow relation, partitioned into synchronous flows* L^S *and asynchronous flows* L^A*.*
- $I \subseteq (E^T \cup E^E) \times E^S$ *is a trigger relation, partitioned into synchronous triggers* I^S *and asynchronous triggers* I^A*.*
- $\chi \subseteq (E \times E) \times (E \times E)$ *is a conflict relation, indicating flows that are mutually exclusive where* $(e, e_1), (e, e_2) \in \{L \cup I\}$ *or* $(e_1, e), (e_2, e) \in \{L \cup I\}$
- $\mu : E \to \mathcal{P}(\mathbb{N}_0)$ *denotes the multiplicity set of an event.*
- $\pm \subseteq (E^T \times E^C) \cup (E^C \times E^T)$ *is the correspondence relation between events of the same process, demanding they send respectively receive the same number of messages* ◇

Figure 15 shows an exemplary BPA, the "BPA construction permit" that we use to explain different concepts of our BPA definition. The BPA consists of five processes $p_1, ..., p_5$. p_1 depicts the "construction permit application" process by an architect, the "construction permit examination" process p_2, executed by the building authority, the "create expert report" process p_3 by experts, and the different surveillance processes p_4 and p_5 performed by the engineers from the building control department. In this scenario, an architect applies for a building construction permit that is examined by the building authority. Depending on the type of construction, it requires several expert reports. In between, the building authority may require further information on the construction project from the architect. After receipt of the expert reports, it informs the architect and orders the according type of control of the construction project depicted by the conflict relation symbol.

The conflict relation χ relates different flows from one event e which exclude each other. Assume sending event e has three flows (e, e_1), $(e, e_2) \in L, (e, e_3) \in I$ and $(e, e_1) \chi (e, e_2)$. Then it sends a trigger signal to e_3 and a message to either e_1 or e_2. Similarly, we can describe that the flows leading to the same receiving event are in conflict. Assume the start event e_s has two incoming triggers from (e_1, e_s), $(e_2, e_s) \in I$ then the process can be triggered by two different processes but always only one process triggers it. We visualize the conflict relation χ similar to a round XOR shape as known from EPCs.

The multiplicity set μ contains all valid numbers of messages or trigger signals an event can send or receive. $\mu(e) = \{1\}$ is called *trivial* and is omitted in graphical representation.

In some case, we would like to express that one event receives the same number of information flows that another event has produced. This can be expressed with the correspondence relation $\pm$. The correspondence relation $\pm$ relates sending and receiving intermediate events of the same process to each other. It requires that the receiving event receives the same amount of messages as the related sending event, or vice versa, that it sends as many replies as it has received messages before. For instance, in the "BPA construction permit" scenario in Figure 15, the building authority, process p_2, orders three expert reports from external experts, process p_3, which it expects to receive for making a decision on the building construction application. It can only continue when it has received all three expert reports and the intermediate sending and receiving events of the permit examination process would be in correspondence relation. In other cases the building authority may require only two of three reports in which the correspondence relation would not be used.

For every event we can define the preset $\bullet e = \{e' \in E^E \cup E^T \mid (e', e) \in I \cup L\}$ of e contains the events with an outgoing relation to $e \in E$. The set $e\bullet = \{e' \in E^S \cup E^C \mid (e, e') \in I \cup L\}$, called *postset of* e, consists of the events with an incoming relation from $e \in E$.

Figure 15 shows an exemplary BPA. Business processes are graphically depicted as rectangular shapes with rounded corners. All events are of triangular shape. Receiving events point into the process, sending events point outwards. Start events consist of only a blank triangular shape. End events are depicted by a triangular shape completely filled with black. Catching intermediate events are represented with two blank layered triangles. Throwing intermediate events use the same symbol like catching intermediate events, except that the inner triangle is filled with black. The graphical representation of business processes is read from left to right, it indicates a time order. An event of a process that has a neighboring event on its right side occurs before that event. Start events are always located on the left side of the process shape and end events on the right side of the process shape. Catching intermediate and throwing intermediate events are located on the top and lower sides of the business process shape indicating that they occur during process execution.

Trigger relations always end in a start event and begin with any sending event. They are represented with black continuous arrows. Information flow relations always end in a catching intermediate event and begin with any sending event. In the graphical representation they are represented as dotted arrows. Synchronous flows are depicted with blank arrow heads, and asynchronous with black filled arrow heads respectively. The χ relation is depicted as round symbol with an x inside. The source event points to the χ symbol that points to the events that are exclusive to each other as depicted in Figure 15. It is inspired by the XOR-gateway in BPMN or EPC. The $\pm$ relation is not depicted in the graphical representation

The BPA depicted in Figure 15 consists of five business processes p_1, p_2, p_3, p_4, and $p_5 \in V$. E. g., business process $p_1 = \{e_1, e_2, \dots, e_8\}$ consists of eight events. In total the BPA depicted has start events e_1, $e_7, e_{13}, e_{15}, e_{17} \in E^S$, end events $e_6, e_{12}, e_{16}, e_{18} \in E^E$, and intermediate events e_2,...,e_5,$e_8 \dots e_{11} \in E^T \cup E^C$. For instance, intermediate events e_8, and e_{11} have a multiplicity set of $m(e) = \{2,3,4,5\}$ that represents the sending and receiving of several information objects. The architecture depicts four information flows and four triggers connecting the sending and receiving events of the different processes. The semantics of the Business Process Architecture depicted in Figure 15 is described in Section 4.5.

BPA COMPENDIUM AND BPA SUBSETS In the previous section, we defined a general BPA that describes all processes of an organization and their interdependencies. However, this concept does not yet provide a structure for a process model collection. To provide a simple hierarchical structure for process model repositories and its inherent process models we introduce three repository layers for our BPA approach, depicted in Figure 16. On the top-layer of a business process repository we situate the *BPA compendium*, on the middle layer we locate *BPA subsets*, and the lowest layer consists of the detailed business process models. The concepts of *BPA subset* and *BPA compendium* will be explained in the following.

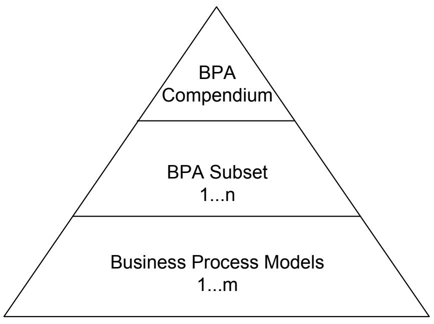


Figure 16: BPA repository structure

In an organization not all processes are interconnected with each other. Only a subset of an organization's processes is involved in handling a specific external request. Such subset of interconnected processes is responsible for a group of related external requests and internally realizes the delivery of a service or the assembly of a product of an organization. For example, the scenario of applying for a construction permit is handled by the interplay of five processes depicted in Figure 15. We address this issue with the concept of BPA subsets. A BPA subset is a non-disjoint subset of processes responsible for handling a business scenario. In regard to the repository structure, the *BPA subsets* are the first grouping level. On a lower level, process models describe the control flow of business processes in a detailed and explicit way. A BPA subset can be considered a BPA itself. A BPA subsets however cannot have any further subsets and all processes of the subset must be somehow interconnected.

Hence, the overall BPA of an organization or company is the set of all BPA subsets whereas all BPA subsets are disjoint. We call the overall BPA, the *BPA Compendium* of an organization. Figure 17 shows an exemplary BPA compendium of a public organization. It resembles a process map except from the fact that all processes in the BPA subset are truly interconnected. It provides the top-layer of our BPA structure.

Public services or the production of a product are sometimes just realized by a single process. In this case, the according business process is independent from all other processes in the process model collec-

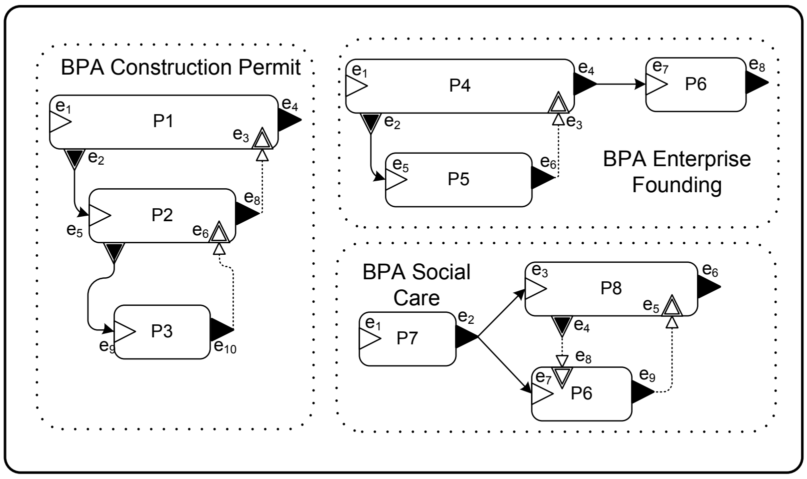


Figure 17: BPA compendium with BPA subsets

tion. A process that is not in a trigger or information flow relation to any other process in the process model collection, i. e., the process is independent, is called a *minimal* BPA, and in this case also considered a BPA subset. Business processes of one BPA subset cannot be in a trigger or information flow relation with business processes of any other subset. If a process has an interdependency with another process then they must reside in the same BPA subset.

Definition 30 (Business Process Architecture Compendium) *A business process architecture* BPA *consists of BPA subsets* $\mathsf{BPA}_1, \ldots, \mathsf{BPA}_n$*, such that*

- $\mathsf{BPA} \subseteq \bigcup_{i=1}^{n} \mathsf{BPA}_i$ *is the BPA compendium*
- $\mathsf{BPA}_1, \ldots, \mathsf{BPA}_n$ *are a parts of the partition of* BPA *representing each a BPA subset.*
- $\mathsf{BPA}_1, \ldots, \mathsf{BPA}_n$ *are pairwise disjoint, i. e., that each process only belongs to one subset.*
- $\nexists(e_i, e_j) \in L \cup I, e_i \in \mathsf{BPA}_i, e_j \in \mathsf{BPA}_j$*, there is no trigger or information flow relation between elements of different subsets.* $\diamond$

Figure 17 shows a complete BPA with its different BPA subsets. The BPA subsets are visually framed with a dotted line and include the name of the BPA subset, e. g., "BPA construction permit", the description of the final result of the process interaction or the scenario that the BPA realizes. Subsets can differ in number of business processes, from a minimal BPA with only one independent business processes to larger BPA subsets that consists of many interdependent business processes. In the following, we will refer to the overall BPA as *BPA compendium*, to *BPA subsets* as *BPA*.

4.5 BUSINESS PROCESS ARCHITECTURE SEMANTICS

In research so far, business processes have been examined one by one and their instantiation and termination have been specified on process model level, e. g., in regard to EPCs in Decker and Mendling [25] or in a specification like BPMN 2.0 [112]. The instantiation of BPAs, representing a whole system of interacting business processes models, raises questions on when such a system starts and terminates, and how the desired behavior of such system looks like.

In Section 4.4 we introduced the main concepts of BPA by defining the BPA elements, and their relations. This section presents the semantics of each BPA element and of BPAs as a system of interdependent processes. The semantics of a BPA define the cooperation between the business processes and the context in which they interact. They determine when a business process can be instantiated, when it completes, at what time which events can throw or catch, and which process depends on which other processes. For this we take a closer look at the occurrence of events, trigger and information flow relations between business processes, and the instantiation and termination of BPAs and their inherent processes. This will serve as base for defining properties and correctness criteria for BPAs

4.5.1 *Event Occurrence*

Events are the basic elements of BPA processes. Their occurrence describes the behavior of a process. An event of a process is either in a trigger or information flow relation or unrelated to other events. This impacts their ability to occur. Start events are ready to occur at any time. Upon occurrence they activate a process instance. Intermediate events are ready to occur for active BPA process instances. An end event is ready to occur for an active instance and terminates that process instance upon occurrence. This implies that all specified intermediate events for that process instance have occurred.

A trigger relation from an end event $e_e \in E^E$ or a throwing intermediate event $e_t \in E^T$ of process p_1 to a start event $e_s \in E^S$ of another process p_2 defines that the event e_e triggers e_s, i. e., start event e_s occurs and instantiates process p_2. If the triggering relation is synchronous, e_e and e_s occur at the same time; if the relation is asynchronous, the event e_e is buffered and can trigger e_s at a later time. The information flow relation from an end event $e_e \in E^E$ or throwing intermediate event $e_t \in E^T$ of one process p_1 to a catching intermediate event $e_c \in E^C$ of another process p_2 expresses that a message is sent from process p_1 to the process p_2. Hence, the information flow relation represents that upon the occurrence of event e information is passed to event e', causing it to occur as well. In both relations the emitting event either portrays the final output of a process (end event) or an intermediary result (throwing intermediate event).

If a sending event takes part in several information flow or trigger relations then it triggers or messages its according partners at the same time. This depicts that one process is started by the sending event e and the other process only requires the same information during process execution. We refer to such communication structure as one form of multi-communication, in this case broadcasting. For example, the occurrence of the event "order ready" signifies that the order is ready for all parties involved, such that the occurrence of one "order ready" event is sufficient to trigger all others. Note, however, that the message that the event has occurred may take some time to convey. In other cases, information flows are buffered and only read upon activation of a process. This depicted, motivates the introduction of asynchronous triggers in Business Process Architectures. In special cases, if the conflict relation is applied, the information that an event occurred is routed only to one of the specified partners.

In contrast to other approaches, BPAs provide means to model *multiplicities*, a term subsuming the sending and receiving of variably many messages to and from multiple process instances of several processes. In the model this is expressed by assigning to each event of the BPA a multiplicity set μ, which indicates how many information flows or triggers the according event sends upon occurrence or requires to occur. For instance, a sending event $e \in E^E \cup E^T$ with $\mu(e) = \{1, 2\}$ expresses that it sends 1 or 2 triggers/information flows. If event e is part of a trigger relation, it sends 1 or 2 triggers to the start event of the receiver process. If the receiving start event has a trivial multiplicity, the process is instantiated either 1 or 2 times respectively.

A receiving event with a multiplicity set of $\mu(e) = \{2\}$ needs to receive two triggers (start event) or two information flows (intermediate catching event) to occur. Events that have zero as one of the elements in their multiplicity set occur optionally. In this regard, start and end events form a special case, further explained in Section 4.5.2.

4.5.2 *BPA Process Instantiation*

A BPA consists of several processes. Before being able to make a statement on the instantiation of a BPA, the instantiation of its processes needs to be defined. The concept of BPA process instantiation partly relies on instantiation concepts of the process model notation. Those, however, are not always clear as stated by Decker and Mendling in [25] for the example of EPCs. In process model collections especially the triggering of other processes is not clearly depicted as the modeling procedures followed, often focus only on single models. In these cases, our BPA approach fills this gap by defining clear trigger relations between processes.

A BPA process can be instantiated by either external stimuli or by receiving a trigger from another process in the BPA. A process which start event does not take part in a trigger relation is triggered by an external stimulus or time, e. g., an order received or an job application submitted or the 15th of the month as start of a regular reporting process. External stimuli describe phenomena that are out of control of the companies operation but are expected events on which a reaction, the business process, is defined. Processes that are triggered by external stimuli do not depend on other processes in the process model collection in regard to their instantiation.

If the BPA process's start event is in a trigger relation with another process's intermediate throwing or end event, then the process is instantiated when it receives a trigger from the other process. In case a BPA process's start event is in trigger relation with several other events then the process is instantiated when it receives one trigger from any of the other processes, unless the event's multiplicity set requires more triggers for instantiation. A start event's multiplicity set specifies the valid options of required triggers. E. g., if a start event has a multiplicity set of $\{2, 5, 7\}$, its process is instantiated once when the start event has received either $2, 5$ or 7 triggers. For processes with start events with trivial multiplicities each received trigger causes the instantiation of one process instance.

A start event can only be assigned a zero to its multiplicity set if it takes part in a trigger or information flow relation. In this case, the zero multiplicity acts similar to the conflict relation, i. e., a start event either needs to receive a trigger to occur or it occurs due to an external stimulus depicted by the zero multiplicity.

4.5.3 *BPA Process Termination*

A BPA process terminates when its end event has occurred. This implies that its start event and all its mandatory intermediate events have occurred. Catching intermediate events that are signaled after termination are not considered by that process instance anymore. Analogous to the start event, end events can only be assigned a zero to its multiplicity set if it takes part in a trigger or information flow relation. In this case the end event optionally sends a trigger or information flow upon its occurence and termination of the process instance.

The BPA process instantiation and termination semantics considering the trigger and information flow relations allow to depict the instantiation semantics of different process modeling notations up to multi-instance concepts.

In the following we summarize the general semantics of BPA processes.

Definition 31 (Semantics of a BPA process and their events [40]) *Processes in a business process architecture* $BPA = (E, V, L, I, \mu, \chi, \pm)$ *must observe the following behavioral contract.*

- *An instance of process* $v \in V$ *must become active when a start event* $e_s \in E^S \cap v$ *occurs.*

- *A start event* $e_s \in E^S$ *is always ready to occur. An intermediate event* $e \in E^T \cup E^C$ *is ready to occur for active instances of* $v \in V$ *for which* $e \in v$*. An end event* $e_e \in E^E$ *is ready to occur for active instances of* $v \in V$ *for which* $e_e \in v$*. This implies that each intermediate event* $e \in v \cap (E^T \cup E^C)$ *has occurred, i. e., it has received the numbers of information flows (catching intermediate event), or emitted information flows and triggers (throwing intermediate event) specified by* $\mu(e)$ *for that instance.*

- *If* $e \in E \wedge \bullet e = \emptyset$ *holds, event* e *can occur for an instance, if it is ready to occur for that instance.*
- *If* $e \in E \wedge \bullet e \neq \emptyset$ *holds, event* e *must occur for an instance if and only if: (i) it is ready to occur for that instance; and (ii) it gets a trigger or information flow.*
- *An instance of process* $v \in V$ *ceases to be active if an end event* $e \in E^E \cap v$ *occurs.*
- *For a synchronous relation* $(e_1, e_2) \in I^S \cup L^S$*, if* e_1 *occurs,* e_2 *must get a trigger or information flow at the same time.*
- *For an asynchronous relation* $(e_1, e_2) \in I^A \cup L^A$*, if* e_1 *occurs,* e_2 *must get trigger or information flow at some time in the future.* ◇

4.5.4 *BPA Instantiation*

The instantiation of a BPA considers the initial start of the system that a BPA represents. A BPA is instantiated if one of its processes is started by an external stimulus, e. g., the desire of a citizen to build a house or a customer's order of a product. I. e., a BPA must have at least one process with a start event that is triggered by an external stimuli or that has a start event containing a zero in its multiplicity set.

In reality, business operations define a range of possible combinations of business processes instantiating a BPA. In this regard, one or several processes may be instantiated by external stimuli. Assume the following BPA scenario of the yearly reporting at a financial service provider, depicted in Figure 18. The yearly reporting is started at the end of the year, but requires quarterly reports that are provided by the quarterly reporting process and half yearly reports that are performed by the half year reporting process. The quarterly process is instantiated on the 15th of each last months of the quarter. The half year processes is instantiated on the 1st of June and 1st of December. Each of these processes is started on different dates but flow into the yearly reporting process. The half-yearly process is dependent on the quarterly reports as it integrates them among other information into the half-yearly report. Similarly, the yearly reporting can only terminate if it has received two half-yearly reports and four quarterly reports. This BPA is instantiated by instantiation of the first quarterly reporting process but eventually all other processes are started as well.

In other scenarios only two of three possible processes need to be instantiated such that the BPA can terminate successfully. For a more specific description of instantiation and termination of a BPA we introduce the concept of *BPA runs*. The instantiation semantics for a BPA run are derived by the BPA scenario and its requirements. To

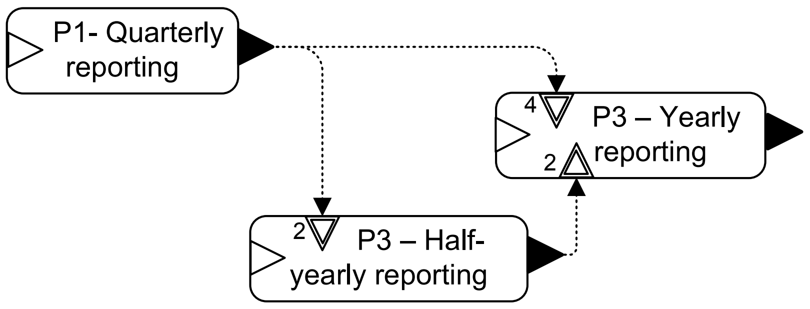


Figure 18: Example of a reporting BPA

specify such complex instantiation semantics the input of a domain expert is required. For this work we assume that we know the different instantiation semantics of a BPA scenario, i. e., which combination of business processes is instantiated by external stimuli. If several processes in a BPA need to be triggered by external stimuli, this may happen in parallel so that all processes are triggered at the same time or eventually so that the processes are triggered one by another eventually. In some scenarios the external stimuli are not synchronized and reach the according processes only one after another as for the example of the yearly reporting BPA scenario.

4.5.5 *BPA Termination*

A BPA instance terminates if all its active process instances have been terminated and no process instances can be activated any further. This means that there needs to be at least one process in the BPA that has an end event with an empty postset or $0 \in \mu$. If a BPA has terminated no triggers or information flows should be sent between any of the business processes.

4.5.6 *BPA Run*

The execution of a BPA is described by a *BPA run* that portrays the instantiation and termination of BPA processes, the occurrence of their events, and the sending of trigger and information flow from the instantiation of a BPA until its termination.

The notion of *BPA* run is defined on instance level. A run of a BPA describes an execution trace of the instantiation, interaction and termination of BPA processes and their inherent events. It also determines how many instances of each process are instantiated during that BPA run. In a BPA run several processes and their according instances may run in parallel or sequentially.

4.5.7 *Initialization of a BPA Run*

A BPA run is initialized with the instantiation of one or several processes that are triggered by external stimuli. All the business processes that are not triggered within the BPA are activated, i. e., all start events that are not part of a trigger relation occur. The external stimuli, for instance, could be the desire of a citizen to build a house as process p_1 in the example in Figure 15. All other start events occur and instantiate their process when they receive the amount of assigned triggers from another process within the same BPA run.

A BPA run determines the multiplicity of each event in the BPA by assigning to each event one element from its multiplicity set. Hence, it is clear for each BPA run how many information flows or triggers each event sends or needs to receive. Depending on the implementation, the multiplicities for one BPA run are either assigned at the creation of a BPA run or at run time. The assignment of multiplicity elements to events in a BPA run, must conform to the correspondence specification ± which is also determined at the creation of the BPA run or at run time.

Events other than start events occur after their preceding events occurred, whereas catching events additionally require to receive the amount of information flows assigned to them by this run. Similarly, end and throwing intermediate events emit the number of triggers or information flows assigned to them by that BPA run.

Not all instances of all BPA processes need to be instantiated in each BPA run. Some BPA runs may exclude processes to be instantiated.

4.5.8 *Termination of a BPA Run*

Each BPA represents the delivery of a service or the production of a good. Its result is the complete fulfillment of a scenario that may have one or several outputs. This is represented by processes which end events do not take part in a trigger or information flow relation or have a zero assigned to their multiplicity set. A BPA run terminates if all active process instances that are part of the BPA run have terminated.

After a BPA run has been initialized, all process instances that were instantiated during the BPA run must terminate, i. e., all their intermediate throwing and catching events, and their end events occurred. In some cases, a BPA may observe lost or remaining information flows and triggers that are not required by any process instance to terminate its execution. A BPA may have many different terminating runs.

A BPA run does not need to instantiate all business processes of the BPA, however, if it does the run is called *covering*. If a process is instantiated by all terminating runs, it is called *essential*.

Building on the concept of BPA runs we introduce BPA properties, namely correctness criteria. The transformation of BPAs into the Open net formalism in Section 5.2 comes with two advantages, the description of clear behavioral BPA semantics as well as the possibility to use well-established analysis techniques.

4.6 BPA PROPERTIES

We define BPA properties to be able to make statements on the structural and behavioral quality of a BPA with focus on the process interdependencies depicted. The BPA properties are grouped into structural properties and behavioral properties similar to properties on process model level like the structural property structural soundness, or the behavioral property Workflow net soundness [165, 146].

Even though, BPAs consist of many interdependent processes, we cannot directly apply the properties for process models at BPA level. Realizability or enforceability (see page 52), properties for process choreographies could be applied but aim at a different use case and hence do not match the scenario of BPAs. Especially enforceability is a rather strict criterion that demands that the initiator of a message exchange must have participated in the previous message exchange as sender or receiver. This is necessary for inter-organizational process interaction but not for intra-organizational process interdependencies where control is within the organization. BPAs have a larger scope than single process models or process choreographies and processes interact in an interleaving way. In this regard, the existing properties for process models are too restrictive to be applied for BPAs. Adapted and new properties for characterizing a BPA's quality are needed.

Based on the BPA properties we design BPA structural and behavioral correctness criteria in Section 4.7 that BPAs should comply to. Section 6.1 and Section 6.2 describe techniques for analyzing BPAs in regard to their structural and behavioral properties.

4.6.1 *Structural BPA Properties*

Structurally, a BPA should always consist of at least one process with independent start event or zero multiplicity and of at least one process with independent end event or zero multiplicity. If there exist no process in a BPA with an independent start event the BPA could never be initialized as no internal or external stimuli could trigger it.

Similarly, a BPA that does not contain a process with independent end event would never result into a final product as the outputs of the processes would be always an input to another process. In this regard, the production chain would never be finished. In some cases this appears to be useful on first sight, e. g., for re-occurring processes.

However, also the output of re-occurring processes should be a well-defined product and not be re-processed forever.

All intermediate events of processes in a BPA should be in a information flow or trigger relation. Intermediate events that take not part in a information flow or trigger relation would either have undefined recipients in case of throwing intermediate events or undefined sources in case of intermediate catching events. Unconnected intermediate catching events would never occur. The emitted information from unconnected intermediate throwing event would get lost as it does not have any receiver. Note, that we assume that the public processes of external partners are available in the process model collection, such that aforementioned scenario should not occur without causing an error.

By definition events are not reflexive, i. e., they should not be put in relation with themselves. Also events of the same type in regard to sending and receiving should not be in relation with each other.

Definition 32 (Well-formed Business Process Architecture) *A BPA is well-formed if it exposes the following structural properties:*

- $\exists\ v \in \mathrm{BPA}$ *with* $e \in E^S \wedge (x, e) \notin (I) \vee \exists v \in \mathrm{BPA}$ *with* $e \in E^S \wedge (x, e) \in (I) \wedge 0 \in \mu(e)$, *i. e., there exists at least one process with an independent start event.*
- $\exists\ v \in \mathrm{BPA}$ *with* $e \in E^E \wedge (e, x) \notin (I \cup L) \vee \exists v \in \mathrm{BPA}$ *with* $e \in E^E \wedge (e, x) \in (I \cup L) \wedge 0 \in \mu(e)$, *i. e., there exists at least one process with an independent end event.*
- $\forall e \in E^T \exists e' \in E^S \cup E^C$, *so that* $(e, e') \in (I \cup L)$, *i. e., there are no unconnected throwing intermediate events so that all intermediate throwing events of all processes are in a trigger or information flow relation.*
- $\forall e \in E^C \exists e' \in E^T \cup E^E$, *so that* $(e', e) \in L$, *i. e., there are no unconnected intermediate catching events so that all intermediate catching events of all processes are in a information flow relation.*

A BPA is well-formed if and only if all above properties hold. ◇

4.6.2 *Behavioral BPA Properties*

For describing the behavioral characteristics and the quality of a BPA we introduce the notion of lost trigger, lost flow, dead events, deadlock, livelock, dead processes, terminating processes, and (lazy) terminating runs.

DEAD EVENT. A dead event is an event that can never occur in any BPA run, i. e., the event is never activated by a trigger or information flow from another event. A start event or intermediate catching event is dead, if it is in a trigger relation but can never occur because their partner process never passes on the trigger. An intermediate throwing event is dead, if the process is never instantiated or the process deadlocks before the occurrence of the intermediate throwing event. An end event is dead if it can never occur, i. e., that the process is never instantiated or never terminates.

LOST TRIGGER/INFORMATION FLOW. A lost trigger or information flow describes the losing of a trigger or information flow in synchronous and asynchronous environments. The trigger or information flow is lost either when it gets emitted in a synchronous environment at a moment at which the target event is not ready, or it gets emitted in an asynchronous environment to an event that may never become ready. We consider unconnected throwing intermediate events causing lost trigger or information flows as they do not have any receiver specified for their information object.

DEAD PROCESS. A dead process is a process that never gets instantiated in any BPA run. E. g., a process never gets instantiated as it requires two triggers to be instantiated but in all possible BPA runs it receives only one trigger.

PROCESS DEADLOCK. A process instance is in a deadlock, if it is active, has one or more end events, and none of its end events can occur at any moment in the future. E. g., a process deadlocks if it has a catching intermediate event that never gets served by another process due to the partner process being dead.

PROCESS LIVELOCK. A process is in a livelock if it triggers itself, so that it creates continuously new process instances without being able to stop triggering itself. This means that the end event does not have a zero multiplicity, is not part of a conflict relation, and only in trigger relation with its own start event. We adapted the usual definition of livelocks from the process model level where a livelock constitutes a state M' from which a particular state M'' can be reached at any time in the future again.

BPA RUN LIVELOCK. A BPA run is in a livelock, if it is in a state, from which it is not possible to reach a state in which all its processes cannot become active anymore. A livelock may be caused by looping behavior where different processes re-instantiate each other without being able to leave that re-instantiation loop. Such a loop may stretch over several processes. A BPA run in a *livelock* does never terminate due to business processes triggering each other in a cyclic fashion. If all BPA runs contain a livelock, then the BPA is *livelocked*.

BPA RUN DEADLOCK. A BPA run is in a deadlock, if it is in a state, where there are still active process instances, but no further communication between the process instances is possible, and the BPA run has not terminated yet. This describes a state, from which it is not possible to reach a state in which all process instances that have been instantiated, terminate. In other words the BPA fails to terminate, if for a process of a BPA one or more occurrences of its start event are observed in a BPA run but the process's end event is not. Such a BPA run is in a *deadlock*. If all BPA runs deadlock, the BPA is called *deadlocked*.

4.7 BPA CORRECTNESS CRITERIA

As Business Process Architectures depict the interplay and interdependencies of several processes, we cannot apply correctness criteria used on individual processes. E. g., the notion of soundness that was introduced for single processes, is too restrictive, as it requires a particular process structure and a very strict control flow behavior. Similarly, although less restrictive, the notion of weak soundness for workflow modules [91], cannot be applied for BPAs as it requires a single start and single end place and does not cover multi-communication and multi-instance concepts found in BPAs. Following the general idea of these properties, we define the following BPA correctness criteria to decide whether a given BPA is correct.

TERMINATING RUN. A BPA run is called *terminating* if it guarantees for all processes, that the end event of a process occurs eventually once its start event has occurred. Hence, in a terminating run all processes that are instantiated also terminate.

LAZY TERMINATING RUN. The weaker notion of *lazy termination* allows BPA runs with pending messages or left-behind process instances, if at least one instance of every process, which was instantiated by a run, terminates. This allows for deadlocked process instances as long as the overall BPA can terminate. This property is comparable to the Workflow net properties *relaxed soundness* and *lazy soundness* [118, 156]. Relaxed soundness was introduced to match more real world scenarios of business processes and allows for deadlocks in some process instances as long each transition participates in a sound firing sequence. The overall correctness of a BPA is determined by examining all BPA runs.

Definition 33 (Correctness Criteria for BPAs (BPA subsets)) *A BPA is correct if it complies to the following rules:*

- *The BPA has at least one (lazy) terminating run*
- *The BPA is free from dead processes*

- *The BPA is not deadlocked*
- *The BPA is not livelocked*
- *The BPA is well formed*

A BPA compendium is correct if all its BPAs (BPA subsets) are correct. ⋄

A BPA with at least one terminating run is called *terminating*. If all runs are terminating we call the BPA *complete terminating*.

A BPA with at least one lazy terminating run, is called *lazy terminating*. If all BPA run are lazy terminating, the BPA is called *complete lazy terminating*.

4.8 SUMMARY

This chapter introduced the basic foundations of our BPA approach. It is characterized by its focus on information flow and trigger relations between business processes. The core idea is to provide a framework to depict the journey through an organization's business processes defined by their interdependencies. For instance, by that departmental barriers are crossed and an end-to-end process view is created. This leverages the information on process interdependencies on a more abstract level and hence is easier to grasp for users of a process model repository. This holistic perspective can be enriched with additional process meta-data for reasoning on overall cost, time, or resources for example. In contrast to our approach, most of the current PA approaches classify single processes based on their functions, goals, or objects and ignore their interdependencies (see Section 3.2). This in return only allows reasoning on one single process at a time.

In many cases business processes are only modeled in isolation and do not directly visualize any interdependency with other process models on model level. Only by taking a holistic view with our BPA approach, interdependencies between processes can be discovered and visualized. This view is taken to some extent by choreography and service composition approaches in regard to inter-organizational process interaction and service composition on process model level. For instance, the research by Decker et al. [24, 28] predominately deals with process choreographies that allow the specification of message exchanges between several actors (participants in a choreography) but hide the involved processes. Barros et al. [10] and Aalst et al. [155] provide an extensive description and analysis on service interaction in cross-organizational workflows based on Petri nets. Their main focus lies on the service composition and refinement with the aim to find matching and compatible services as noted in Chapter 3. These approaches provide a high level of detail. In contrast, our BPA approach portrays process interdependencies on an abstract level, allows specifying multi-communication, and provides a holistic business process end-to-end view.

The formal definition of our BPA approach provides the basis for the development of transformation methods and the design of BPA analysis techniques in the following chapters. Our BPA approach is the first PA approach that is based on a formal definition. It implies three repository layers, the BPA compendium layer, the BPA subset layer, and the process model layer. Chapter 5 defines clear behavioral semantics of BPAs by their transformation to Open nets. The ON transformation and a pattern/ anti-pattern approach are used for the analysis of BPAs regarding their behavioral and structural properties in Chapter 6. Based on our BPA formalism, we design a bottom-up extraction and top-down decomposition algorithm, and consistency criteria between process model level and BPA level in Chapter 7. In Chapter 8 the formalism is used to develop a technique to extract and depict data dependencies between business processes.

5 BUSINESS PROCESS ARCHITECTURE BEHAVIOR

This chapter is based on the published papers [40, 38, 45, 44]. It introduces a transformation of BPA to Open nets. The ON representation of a BPA describes its behavioral semantic in a clear unambiguous way. The chapter provides a central part of our BPA concept. It is the basis for the behavioral analysis of BPAs. For the definition of ONs and their composition we refer to Section 2.5.2. The definition of BPAs was introduced in Definition 29.

The presented BPA framework leaves room for interpretation in regard to their behavioral semantics. Based on the current definition, a strong analysis in regard to presented BPA properties is difficult. To avoid ambiguities in interpretations and facilitate analysis, we transform BPAs to Open nets (ON), a subclass of Petri nets that have clear semantics [110, 148, 32, 80, 165]. Beside providing clear semantics, this transformation allows for analyzing BPAs with known Petri net analysis techniques for their structural and behavioral properties.

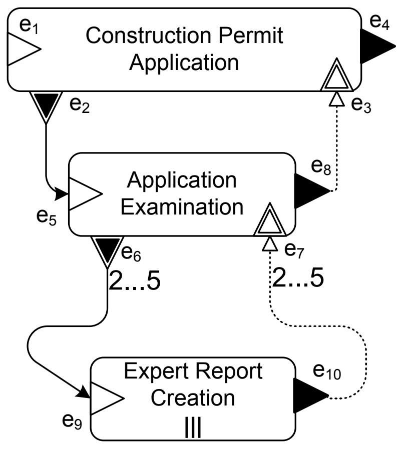


Figure 19: BPA for a construction permit application

For the transformation from BPAs to Open nets we use the BPA definition from Definition 29. We consider BPA business processes to be sequences that have one start, one end event, and any number of intermediate events in regard to their structural composition. In regard to behavioral aspects, this means that a start event always occurs before all intermediate events of a process, and the end event after all intermediates events have occurred, respectively. Events of one process cannot occur in parallel. For the transformation the multiplicity of events of BPA process play an important role. They are used to depict the repeated execution of processes as well as multi-communication. The term multi-communication captures interaction between multiple instances of several processes instead of one-to-one correspondence between instances.

To illustrate our BPA to Open net transformation algorithm and how we deal with multi-communication, we resort to a simplified version of our example from the public administration that depicts an application for a construction permit as BPA, presented in Figure 19. The delivery of public services often involves the interplay of multiple interacting process instances. E. g., an architect sends a construction permit application to the building authority. This message triggers the "construction permit examination" process. Depending on the type of construction, the application is forwarded to between two and five experts instantiating an appropriate number of "create expert report" processes. On termination each instance returns a message to the "construction permit examination" instance, that waits for the according number of messages, then terminates and returns the decision of the building authority to the applicant. In this example multiple instances of the "create expert report" process are triggered and the "construction permit examination" process sends and receives multiple messages.

Such behavior so far was considered only in few other approaches, if at all. However, it is desirable to show the exact behavior of the multiplicities and the depicted multi-communication, and analyze it according to the correctness criteria established. For a BPA that involves multiple instances of processes to work, adjustment of the amount of process instances participating in a run is needed. With the use of multiplicities the required resources in terms of process instances or information sent can be considered and analyzed as well. This forms an asset for the planning of resources, e. g., human resources needed when running multiple instances of a process in a BPA.

Business process modeling approaches that allow to express these types of multiplicity do not offer formal analysis. Formal methods based on Petri nets have been successfully applied to model and analyze workflows (Workflow nets [147, 161]), services and their composition (service or open nets e.g. [94]) as well as process choreographies (public to private approach [150, 157]). However, those elaborated analysis methods do not explicitly deal with multiple instances of processes.

By providing a transformation into ONs [95], which have been successfully applied to study the composition of services and its correctness, we fill this gap and provide a technique to analyze BPAs with multiplicities.

We introduce intermediary nets to represent and analyze multiple instances and multi-communication in the ON formalism. In this way, our technique may also be applied to other use cases than BPAs that involve the modeling and the analysis of multi-communication and multiple instances in ONs.

5.1 BPA MULTIPLICITY CONCEPTS

Business Process Architectures exhibit two kinds of multiplicity: a) multiple instances of a business process and b) sending and receiving multiple messages or trigger signals to and from several other process instances.

MULTIPLE PROCESS INSTANCES. In a BPA the processes that can be instantiated several times are visually indicated by three vertical bars referring to the similar symbol in BPMN. Generally the instantiation of a process depends on the multiplicity set of its start event, the number of different events that are in trigger relation with the start event, and their multiplicity sets. Multiple instances of a process are instantiated if the partner event's multiplicity set contains elements that are a multiple of elements of the multiplicity set of the process's start event, e. g., in Figure 19 the process of the building authority "construction permit examination" can send two to five triggers to the "create expert report" process, instantiating it two to five times. In this example the start event of the "create expert report" process has a trivial multiplicity, and there exists only one event that triggers it. The intermediate throwing event has a multiplicity set with the elements two to five. The concrete number of times a process is instantiated can vary between BPA runs in this case.

The number of process instances instantiated depends on the number of triggers its start event receives, compared to the multiplicity of its start event, as assigned by this particular run. For start events with trivial multiplicity set, each received trigger corresponds to one process instantiation. However, if the multiplicity set of a start event is non-trivial, then the process is only instantiated, if the sum of triggers received from its partners is equal or greater than the multiplicity element assigned to it in the according BPA run. Each trigger from one of the predecessors counts toward the total number of required triggers.

SENDING AND RECEIVING MULTIPLE MESSAGES. Multi-communication encompasses two cases and combinations thereof: the sending or receiving of information flows or triggers to respectively from different partners, and the sending or receiving of multiple information flows or triggers by one event defined by its multiplicity set.

Throwing events can send messages to multiple receiving processes, while catching events can receive messages from multiple sending processes according to the multiplicity assigned to them. In the first case the same amount of messages is delivered to each receiver, while in the second case messages from various senders are collected before being consumed according to the multiplicity specification.

Zero is a valid value in the multiplicity set of a throwing or catching intermediate event, meaning that an information flow or trigger is not sent at all or not required to proceed respectively. The zero in a multiplicity set depicts the optional sending or receiving of an information flow or trigger. The zero as a multiplicity element in the multiplicity sets of start or end events depicts a special case and only appears for start events in a trigger or end events in a trigger or information flow relation. This depicts the special situation that a process

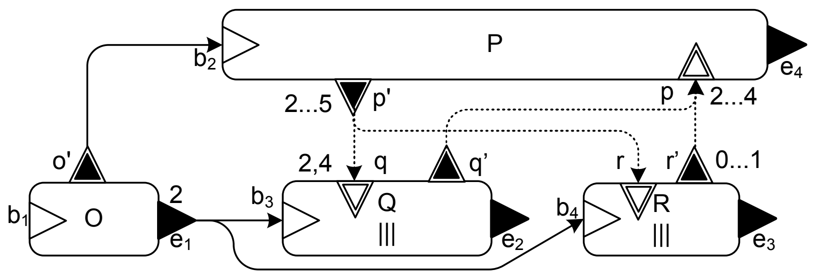


Figure 20: BPA multiplicity concepts

may be started by an external stimulus (depicted by the zero in its multiplicity set) or by another process of the BPA. The zero in a multiplicity set of an end event means that the result of the process is not passed on.

The BPA in Figure 20 illustrates the multi-communication and multiple instances concepts. The example consists of four processes O, P, Q and R. The triggering of multiple instances of several processes is depicted by the triggering relation of O with Q and R respectively where end event e_1 of process O has the multiplicity two. In this way, two instances of each Q and R are instantiated. The receiving of multiple messages is illustrated by the multiplicity of q ($\mu(q) = \{2,4\}$) which means that each instance of Q (of which there are two) waits for either two or four messages to proceed. The concept of optional sending is represented by event r' of process R having the multiplicity $(0 \ldots 1)$. Either it sends the message or not. Collecting of messages takes place in catching event p whose preset contains both q' and r'. p has a multiplicity set of $2 \ldots 4$ and expects between 2 to 4 messages in total from (all instances of) Q and R. Q and R are instantiated each two times and hence deliver the expected amount of information flows for p.

relating event multiplicity specifications. The number of messages a process sends is often closely related to the number it expects to receive, e. g., in Figure 19, if three expert reports are requested, also three results are expected back. This relation between two events is captured in the ±-relation, to which all BPA runs need to conform. The ±-relation reduces the amount of possible BPA runs. A single BPA run assigns to each event an element from its multiplicity set, so that each combination of assignments defines a possible run. Runs contradicting the ±-relation are considered invalid and can be omitted in the state space of a BPA, i. e., the set of all its runs. If the ±-relation is not used, all possible runs are valid and the complete state space has to be explored during analysis.

5.2 TRANSFORMING BUSINESS PROCESS ARCHITECTURES

Due to the definition of ON composition (see Definition 27) we cannot directly express the triggering of varying amount of instances respectively sending or receiving of a varying amount of triggers or information flows with ONs. The case that one event is in trigger or message flow relation with several other events is not directly covered, either. To overcome this we create and insert modular intermediary nets for normal and multi-communication.

The transformation is conducted in a modular fashion: Each BPA process is first independently transformed into an ON. In a second step intermediary ONs are created that capture the trigger and information flow relations and reflect the multiplicity of the participating events. In the last step the processes' ONs and intermediary ONs are composed into one large Open net. The resulting net depicts the behavior of the BPA with inherent multiplicities and can then be analyzed in regard to its properties with model checking tools like LoLA [132].

TRANSFORMING BUSINESS PROCESSES. We first describe the transformation of a single BPA business process into an ON. It is important to note, that all events are unique and only part of one partition.

Definition 34 (BPA Process to ON Transformation) *Given a BPA, let $\langle e_1 e_2 \dots e_n \rangle$ be the sequence of events belonging to the business process $v \in V$ then the process's ON is defined as $O_v = (P_v, T_v, F_v, M_{0_v}, \Omega_v)$, where*

- $T_v = \{t_{e_i} | e_i \in v\}$
- $P_v^N = \{p'_{e_i} | e_i \in v \land 1 \le i < n\} \cup \{p_{e_1} \,|\, \bullet e_1 = \emptyset\} \cup \{p_{e_n} \,|\, e_n \bullet = \emptyset\}$
- $P_v^O = \{p_{e_i} | e_i \in (E^E \cup E^T) \cap v\} \setminus \{p_{e_n} \,|\, e_n \bullet = \emptyset\}$
- $P_v^I = \{p_{e_i} | e_i \in (E^S \cup E^C) \cap v\} \setminus \{p_{e_1} \,|\, \bullet e_1 = \emptyset\}$
- $P_v = P_v^N \cup P_v^O \cup P_v^I$
- $F = \{(t_{e_i}, p'_{e_i}), (p'_{e_i}, t_{e_{i+1}}) \,|\, t_{e_i} \in T_v \land p'_{e_i} \in P_v^N\} \cup \{(t_{e_i}, p_{e_i}) \,|\, t_{e_i} \in T_v \land p_{e_i} \in P_v^O\} \cup \{(p_{e_i}, t_{e_i}) \,|\, t_{e_i} \in T \land p_{e_i} \in P_v^I\}$
- $M_{0_v} = \emptyset$ *if there exists* $(t, e_1) \in I$ *and* $\{p_{e_1}\}$ *otherwise.*
- $\Omega_v = \{\}$ *if there exists* $(e_n, t) \in I \cup L$ *and* $\{p_{e_n}\}$ *otherwise.* ◇

The example in Figure 21 clarifies this rather technical definition. We distinguish between three different types of places: internal places P_v^N, input interface places P_v^I, and output interface places P_v^O. Each event e yields an internal place p'_e, except from the end event. Additionally, each event e produces an interface place p'_e and one transition t_e. For intermediate throwing events this place becomes an input interface place, while for intermediate throwing interface places it becomes

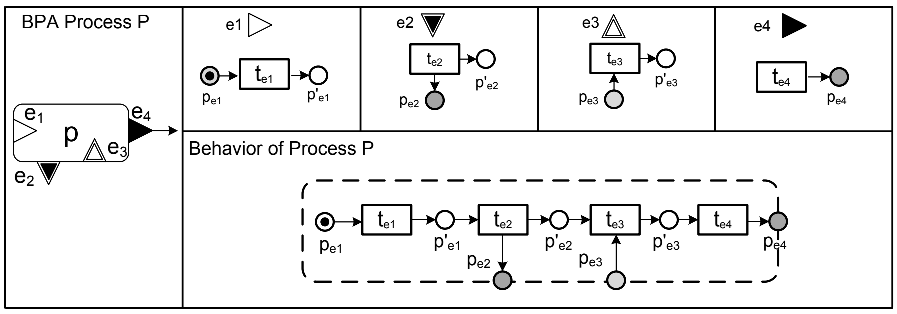


Figure 21: BPA process to ON transformation

an output interface place. For start and end events the situation is different. Depending on the pre- respectively postset the place p_{e1} (for the start event) respectively p_{e_n} (for the end event) can be either an internal place ($e \in P^N$) or interface place ($e \in P^I$) respectively ($e \in P^O$).

The event's transition t_e is connected to its internal place p'_e, which further connects to the transition of the next event in the sequence. Depending on the event type either the transition exposes an outgoing flow to the event's interface place p_e, or an incoming flow from p_e. The direction of the flow depends on the event type. If the event is a start or catching event (light gray in Figure 21) the flow points from place to transition, if it is an end or throwing event (dark gray) it points in the opposite direction. Only if the start event's preset is non-empty $\bullet e_1 \neq \emptyset$, the place p_{e_1} is an input place, otherwise it is an initially marked internal place as is the case in Figure 21. In this case the place p_{e_1} is part of the set of internal places P^N. Equivalently p_{e_n} is an output place only if e_n has a non-empty postset, i. e., it is part of a trigger or information flow relation, otherwise p_{e_n} is an internal place and part of the final marking. In Figure 21 there is an arc leaving e_4 indicating a relation, hence p_{e_n} is an output place and not part of the final marking.

The resulting ON of the BPA process provides a clear behavioral semantics. It is obvious that at first transition t_{e_1} fires, followed by transition t_{e_2}. Transition t_{e_3} can only fire if it also gets input from another process. If transition t_{e_3} gets an input it can fire, else the ON is in a deadlock. After t_{e_3} has fired, transition t_{e_4} is enabled and can fire ending the process.

REPRESENTING MULTIPLE INSTANCES. There are two ways in ONs to represent the BPA concept of multiple instances. Either each instance is represented by a copy of the process's ON, or each instance is represented by a token (colored or black) in the process's ON. Representing each instance by its own open net is problematic when it comes to triggering, as it either requires the creation of nets at runtime or the management of a pool of untriggered nets. Another challenge would be the naming of interface places for each instance net.

Indicating the number of instances as black tokens in one ON eases the composition. In this way, the resulting net has less transitions and places than using one net per instance. Hence we decided to use the multiple black token representation for instances. It is also more compact. Colored tokens would allow to distinguish between cases. However, in the moment we assume one overall correlated case per BPA run, i. e., all processes and instances that participate in BPA run process the same overall case.

Introducing correlation into the BPA concept is an open issue that needs to be examined in the future.

SIMPLE INTERMEDIARY NET. The simple intermediary net is used for trigger or information flow relations in which both events have a trivial multiplicity. In this case a simple intermediary net is created with one input interface place, one transition, and one output interface place. The simple intermediary net is the basic net for all other intermediary nets. It is depicted in Figure 22.

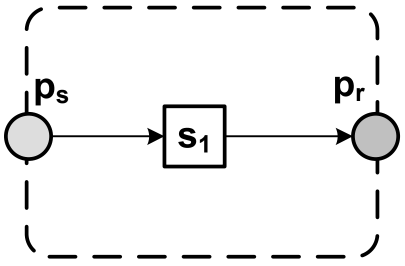


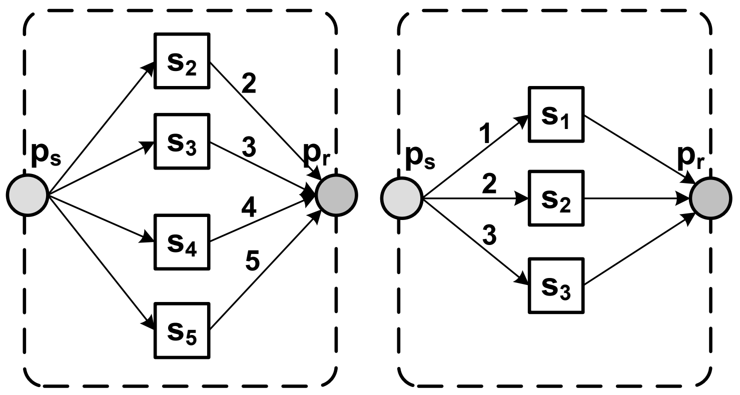
Figure 22: Simple intermediary net

Depending on the type and multiplicity of the events, the net structure is changed to depict multi-communication or conflict constructs accordingly. The simple intermediary net is defined as follows.

Definition 35 (Simple intermediary net) *Given a BPA, the simple intermediary net for a trigger or information flow* $(s, r) \in I \cup L$ *with* $\mu(s) = \mu(r) = \{1\}$ *is defined as* $O_{s,r} = (P_{s,r}, T_{s,r}, F_{s,r}, 0, \{0\})$ *where*

- $P_{s,r} = P^N_{s,r} \cup P^I_{s,r} \cup P^O_{s,r}$ *and* $P^I_{s,r} = \{p_s\}$, $P^O_{s,r} = \{p_r\}, P^N = \varnothing$
- $T_{s,r} = \{s_1\}$
- $F_{s,r}(p_s, s_1) = 1$, $F_{s,r}(s_1, p_r) = 1$. ◇

MULTICAST AND MULTIRECEIVE NET. Depending on the multiplicity of a throwing event it emits a different number of messages or trigger signals. To capture this in the ON formalism, we introduce an intermediary ON called *multicast net*. It is a net schema, because each multiplicity specification entails a different multicast net, consisting of one input interface place, one output interface place, and one transition for each element in the multiplicity set of the event. Note, that the same construct is used for triggering multiple instances of a process as well as sending multiple information flows.



(a) Multicast net (b) Multireceive net

Figure 23: ON intermediary nets for representing the sending resp. receiving multiple information flows or triggers

Definition 36 (Multicast net) *Given a BPA, the multicast net for a trigger or information flow* $(s,r) \in I \cup L$ *is defined as* $O_{s,r} = (P_{s,r}, T_{s,r}, F_{s,r}, 0, \{0\})$ *where*

- $P_{s,r} = P^N_{s,r} \cup P^I_{s,r} \cup P^O_{s,r}$ *and* $P^I_{s,r} = \{p_s\}$, $P^O_{s,r} = \{p_r\}$, $P^N = \emptyset$
- $T_{s,r} = \{s_i \,|\, i \in \mu(s)\}$
- $F_{s,r}(p_s, s_i) = 1$, $F_{s,r}(s_i, p_r) = i$ *where* $i \in \mu(s)$.

The multicast net has the empty multiset as initial and as the only final marking. ◇

E. g., let $s \in E^T$ be a throwing event, $r \in E^C$ a catching event and $(s,r) \in L$ be connected by an information flow. Let further $\mu(s) = \{2,3,4,5\}$ be the multiplicity set of s and $\mu(r) = \{1\}$ be r's set. Then the corresponding multicast net is depicted in Figure 23a, with the input interface place p_s, the output interface place p_r, the four transitions s_2, s_3, s_4, s_5 and the following arcs: p_s is connected to all s_i with arc weight 1 and all s_i are connected to p_r with arc weight i where $i \in \mu(s)$ are the elements of the multiplicity set of s. As a result the multicast net produces between two and five tokens on its output interface place, thus representing the sending of between two and five information flows.

We introduce the *multireceive net*, a slightly adapted version of the multicast net, to express that a process instance waits for a certain number of information flows or triggers before it can continue or that a process is instantiated only after receiving a certain number of trigger signals. The only difference is that arcs from the input place to the transitions now carry the variable weights, while the arc weights between transitions and output place have the value 1.

Definition 37 (Multireceive net) *Given a BPA the multireceive net for a trigger or information flow* $(s, r) \in I \cup L$ *is defined as* $O_{s,r} = (P_{s,r}, T_{s,r}, F_{s,r}, 0, \{0\})$ *where*

- $P_{s,r} = P^N_{s,r} \cup P^I_{s,r} \cup P^O_{s,r}$ *and* $P^I_{s,r} = \{p_s\}$, $P^O_{s,r} = \{p_r\}$
- $P^N = \emptyset$ *for* $0 \notin \mu(r)$ *and* $P^N = p_0$ *for* $0 \in \mu(r)$
- $T_{s,r} = \{s_i \mid i \in \mu(r)\}$
- $F_{s,r}(p_s, s_i) = i \wedge i \neq 0$, $F_{s,r}(s_i, p_r) = 1$ *where* $i \in \mu(s)$

The multireceive net has the empty multiset as initial and as the only final marking. ◇

The resulting intermediary multireceive net construct is depicted in Figure 23b.

The special case of an intermediate receiving event or start event having a zero in their multiplicity set creates an intermediary net with an additional internal place p_0. This place may fire optionally, if marked by a BPA run, so that a process may start or continue its execution without receiving a trigger or information flow from a another process.

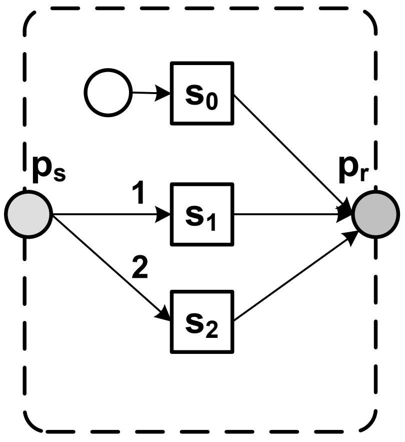
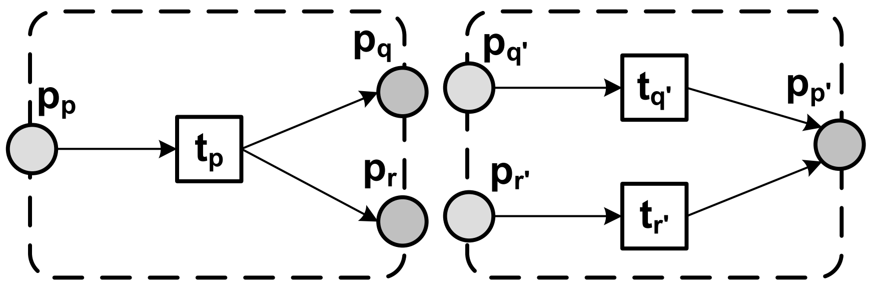


Figure 24: Optional multireceive net

The resulting net schema, illustrated in Figure 24, resembles the standard option of the multireceive net, except from the additional place that depicts the optional occurrence of the receiving event. The number of process instances that are executed without the occurrence of the receiving event is restricted by the amount of tokens put on the internal place p_0 by the BPA run configuration.

SPLITTER AND COLLECTOR NET. Processes can not only trigger multiple instances of one process, but also instances of multiple processes. The same is true for sending and receiving information flows.

We illustrated the different concepts in Figure 20. E.g., process P sends messages to both processes Q and R. Formally we have $p\bullet = \{q, r\}$ two events in the postset of throwing event p. The opposite situation, one process receiving information flows or triggers from several other processes can also be found in Figure 20 where the catching event p' has the preset $\{q', r'\}$. It only matters that the preset respectively postset of an event is non-singleton and not in which relation those events are.



(a) Splitter net for event p (b) Collector net for event p'

Figure 25: Open net constructs for multiple receivers respectively senders

While the splitter net in Figure 25a takes a token from one source and produces one token for each target, the collector net in Figure 25b collects all tokens from several sources in one place. The collector net depicts that a process can be triggered or can get an information flow from different sources. Each information flow or trigger would result in the occurrence of the receiving event, unless the event has a multiplicity larger than 1. The collector net and the receive-conflict net do exhibit the same structure. Depending on the composition of intermediary nets the net schema takes either the role of a collector net or receive-conflict net. Splitter and collector net are also net schemata. Principally, the splitter net has one input place, one transition, and one output place for each start or catching event $b \in E^S \cup E^C$ that is in trigger or information flow relation with a given end or throwing event $t \in E^E \cup E^T$. The collector net has one place and one transition for each event $e \in \bullet e'$ for the receiving event $e' \in E^S \cup E^C$. All arcs have the weight 1.

Definition 38 (Splitter net) *Given a BPA and a sending event $e \in E^T \cup E^E$, the ON $O_e = (P_e, T_e, F_e, 0, \{0\})$ is called the* splitter net for e, *where*

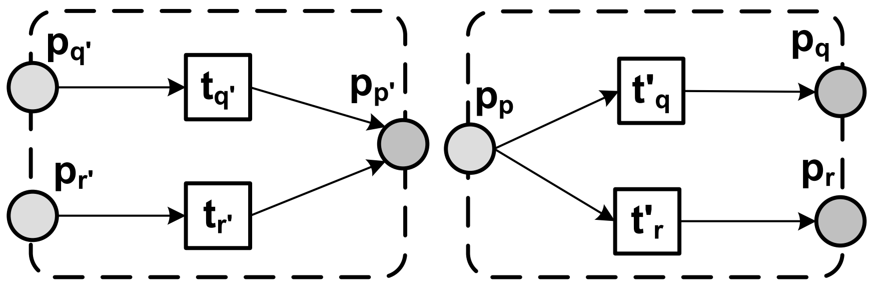
- $P_e = P_e^N \cup P_e^I \cup P_e^O$, $P_e^N = \emptyset$, $P_e^I = \{p_e\}$, $P_e^O = \{p_b \mid b \in e\bullet\}$
- $T_e = \{t_e\}$
- $F_e(p, t_e) = F_e(t_e, p) = 1 \ \forall p \in P_e$

The initial marking is the empty multiset, which is also the only final marking. ◇

Definition 39 (Collector net) *Given a BPA and a receiving event $e' \in E^S \cup E^C$. Then the ON $O_e = (P_e, T_e, F_e, 0, \{0\})$ is called the* collector net for e', *where*

- $P_{e'}^I = \{p_e \mid e \in \bullet e'\}$, $P_{e'}^O = \{p_{e'}\}$ *and* $T_{e'} = \{t_e \mid e \in \bullet e'\}$.
- $F_e(p, t_e) = F_e(t_e, p) = 1 \ \forall p \in P_e$

The initial marking is the empty multiset, which is also the only final marking. ◇



(a) Receive-conflict net (b) Send-conflict net

Figure 26: Open net constructs for receive-conflict and send-conflict net

RECEIVE-CONFLICT AND SEND-CONFLICT NET. Processes do not always send their intermediary or final outputs to all processes they are in relation with. This is the case, if the result is of interest to only one of the partner processes, e. g., due to a data object being in a particular state.

Although structurally identical to the collector net, the receive-conflict net differs in the fact that the receiving event is part of several trigger or information flow relation that are in conflict with each other. For each event that is in conflict with another event in the preset of the receiving event an input interface place and a transition is created. One output interface place is created for the receiving event in the receive-conflict intermediary net. Depending on the position in the composition of intermediary nets the net schema takes either the role of a collector net or receive-conflict net. We explain the composition of intermediary nets in Section 5.3. Both the receive-conflict and send-conflict net are depicted in Figure 26a and Figure 26b.

Definition 40 (Receive-Conflict net) *We denote the incoming conflict set of an event* e' *as* $\#e' := \{e | \exists e_c \in \bullet e' : (e, e'), (e_c, e') \in \chi\}$. *Given a BPA, a receiving event* $e' \in E^S \cup E^C$, *with* $\#e' \neq \emptyset$, *the ON* $O_{e'} = (P'_{e'}, T'_{e'}, F'_{e'}, 0, \{0\})$ *is called the* conflict net for e', *where*

- $P^I_{e'} = \{p_e \,|\, e \in \bullet e' \wedge e \in \#e'\}$, $P^O_{e'} = \{p_{e'}\}$
- $T_{e'} = \{t_e \,|\, e \in \bullet e' \wedge e \in \#e'\}$
- $F_e(p, t_e) = F_e(t_e, p) = 1 \;\forall p \in P_e$

The initial marking is the empty multiset, which is also the only final marking. ◇

In a similar way we create the *Send-Conflict net*. In this case the sending event is part of several trigger or information flows that are in conflict with each other depicted by the relation χ. For each event in the postset of the sending event a transition and interface output place is created whereas only one input interface place is created for the sending event. The send-conflict net is defined as follows.

Definition 41 (Send-Conflict net) *We denote the outgoing conflict set of an event* e *as* $e\# := \{e' | \ \exists e_c \in e\bullet : (e, e'), (e, e_c) \in \chi\}$. *Given a BPA, a sending event* $e \in E^E \cup E^T$ *and* $e' \in e\#$. *Then the ON* $O_e = (P_e, T_e, F_e, 0, \{0\})$ *is called the* conflict net for e, *where*

- $P_e^I = \{p_e\}\ P_e^O = \{p'_e \,|\, e' \in e\bullet \wedge e' \in e\#\}$
- $T_e = \{t'_e \,|\, e' \in e\bullet \wedge e' \in e\#\}$
- $F_e(p, t'_e) = F_e(t'_e, p) = 1\ \forall p \in P_e$

The initial marking is the empty multiset, which is also the only final marking. ◇

5.3 COMPOSITION OF NETS

The presented net constructs enable us to represent BPAs and their different multi-communication constructs as Open nets, clarify their behavior, and analyze them. After the transformation of BPA processes, event multiplicities, and flow relations the resulting ONs and intermediary ONs have to be composed according to Definition 27. The composition relies on matching interface places. Because each event is unique in a BPA and each event is represented by at most one interface place, those interface places are unique too. For each pair of events in a trigger or information flow relation at least one intermediary net is created to connect the according ONs. Those are defined to provide the complementary interface places and make the nets composable. Without the intermediary nets, the composition would yield unconnected nets leaving all places unfused.

The set of four basic composition rules describes the composition of ONs by introducing simple intermediary nets for events with trivial multiplicity that are part of only one trigger or information flow relation. Composition rules 5-8 depict the composition of ONs for conflict and multi-communication constructs. The complex composition rules describe the composition of ONs by introducing a combination of multi-communication intermediary and conflict nets for events with non-trivial multiplicity and events that are part of many trigger or information flow relations.

Figure 27 shows the transformation of the basic trigger and information flow relations between two BPA processes.

Composition rule 1 maps the trigger relation $(e_1, s_2) \in I$ from an end event of process p to a start event of another process q. According to Definition 34 and Definition 35 the transformed ONs of process p and process q are composed via a simple intermediary net that connects the end place of ON_p to the start place of ON_q. The initial place of ON_p is marked with a token as its origin event s_1 has an empty preset whereas the former initial place of ON_q remains empty.

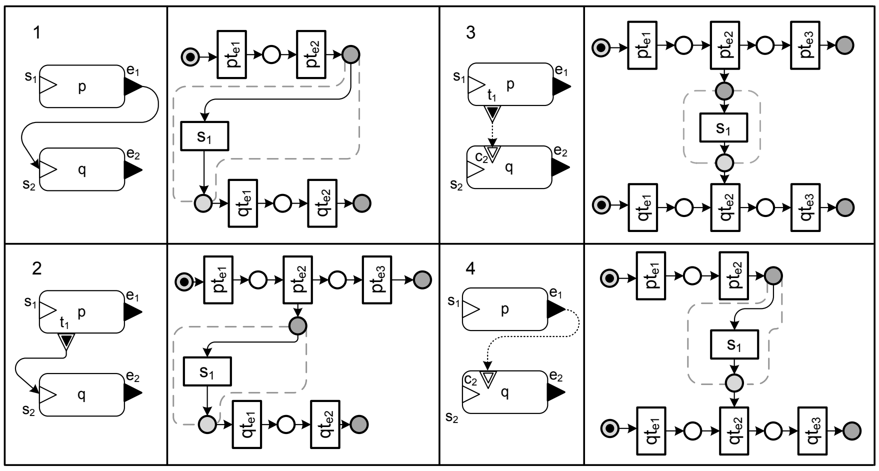


Figure 27: Basic composition rules with simple intermediary net

The token will be passed on by ON_p when it triggers ON_q. We notice this kind of relation from end to start event results in a structurally and behaviorally sound ON.

Composition rule 2 describes trigger relation $(t_1, s_2) \in I$. Process p triggers process r through its intermediate event t_1. To represent this trigger relation both processes are connected via a simple intermediary net that connects the output interface place of transition pt_{e2} of ON_p to the initial input interface place of ON_q. The resulting ON has one marked initial place and two end places.

Composition rules 3 and 4 describe the information flow relation by sending and catching an information flow through intermediate events in rule 3 as well as by passing an information flow through an end event to an intermediate event in rule 4, respectively. In rule 3 the information flow relation $(t_1, c_2) \in L$ is presented by the connection of the output interface place of transition pt_{e_2} via a simple intermediary net to the input interface place of transition qt_{e_2} of ON_q. The composed ON has two initial and two end places. Composition rule 4 is similar, except from the source of the composition being the end place of ON_p. The resulting ON has two initial places and only one end place. Note, that transitions pt_{e_1} and qt_{e_1} in rule 3, 4, and 6 are not synchronized and can fire independently.

Composition rules five to eight, shown in Figure 28, depict multi-communication and conflict relations between several processes. The events involved have trivial multiplicity. The composition are examples for all different kinds of combinations of information flow and trigger relations between processes.

Composition rule 5 defines the triggering or messaging of several processes by one process, i. e., an event takes part as source in several trigger or information flow relations. Depicted in Figure 28, process p triggers process q as well as process r. The end event e_1 of process p takes part in two trigger relation pairs (e_1, s_2) and (e_1, s_3). In

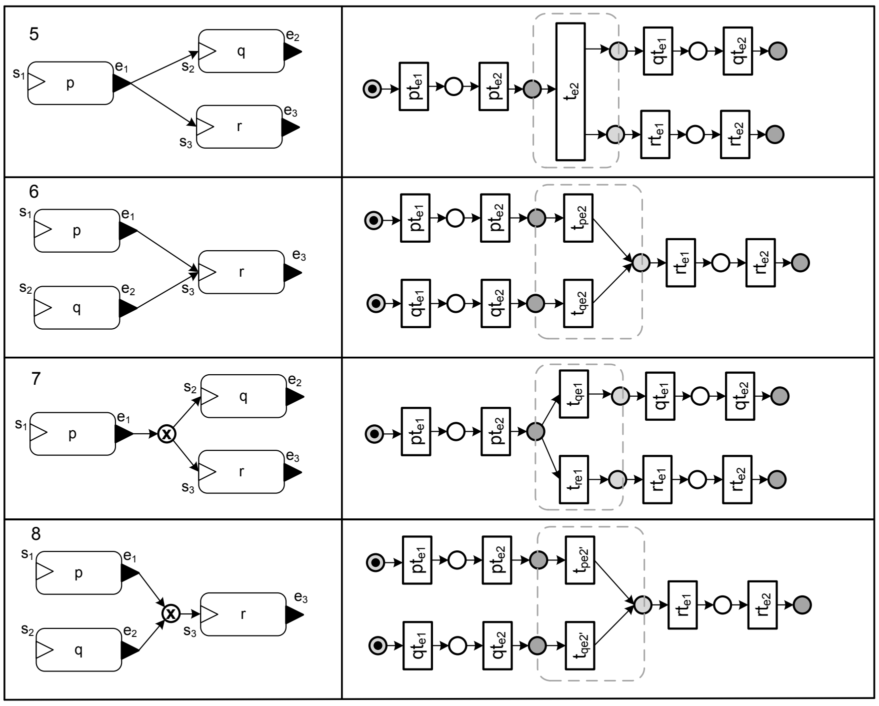


Figure 28: Multi-communication composition rules with events of trivial multiplicity

this case, ON_p is connected to ON_q and ON_r through an intermediary splitter net. When transition t_{e2} fires, it passes on tokens to two concurrent branches, former ON_q and ON_r.

Composition rule 6 describes the receiving of several triggers or information flows, i. e., an event takes part as destination event in several trigger or information flow relations. In Figure 28, event s_3 takes part in $(e_1, s_3) \in I$ and $(e_2, s_3) \in I$. The three ONs are connected through an intermediary collector net. The intermediary collector nets input interface places are connected to the output places of ON_p and ON_q. The output interface place is connected to the interface input place of ON_r resulting in one composed ON. ON_r can be triggered by ON_p and ON_q or by both, one after the other.

Composition rule 7 describes the one option of the conflict relation between several flows, i. e., an event is part of several trigger or information flow relations that are in conflict. The rule depicted here shows that the event e_1 is part of two triggers that are in conflict with each other, so that $((e_1, s_2), (e_1, s_3)) \in \chi$. If two flows are in a conflict a transition for each relation in χ is created and connected. Instead of connecting all processes via an intermediary splitter net, we introduce an intermediary send-conflict net that represents the conflict $((e_1, s_2), (e_1, s_3)) \in \chi$ between those flows. Rule 7 is a refinement of composition rule 5.

Composition rule 8 depicts the same structural intermediary net as the collector net. For events with trivial multiplicities there is no structural difference. It depicts the receive-conflict relation between two flows in a BPA. s_3 is part of two triggers that are in conflict with each other, so that $((e_1, s_3), (e_2, s_3)) \in \chi$.

COMPLEX COMBINATIONS OF INTERMEDIARY NETS. In complex cases additional care has to be taken to avoid wrong composition. This is required for events that have both a non-trivial multiplicity set $|\mu(e)| > 1$ *and* a non-singleton postset $|e\bullet| > 1$. The combination of intermediary nets needs to follow a particular order which we describe in composition rules nine to eleven.

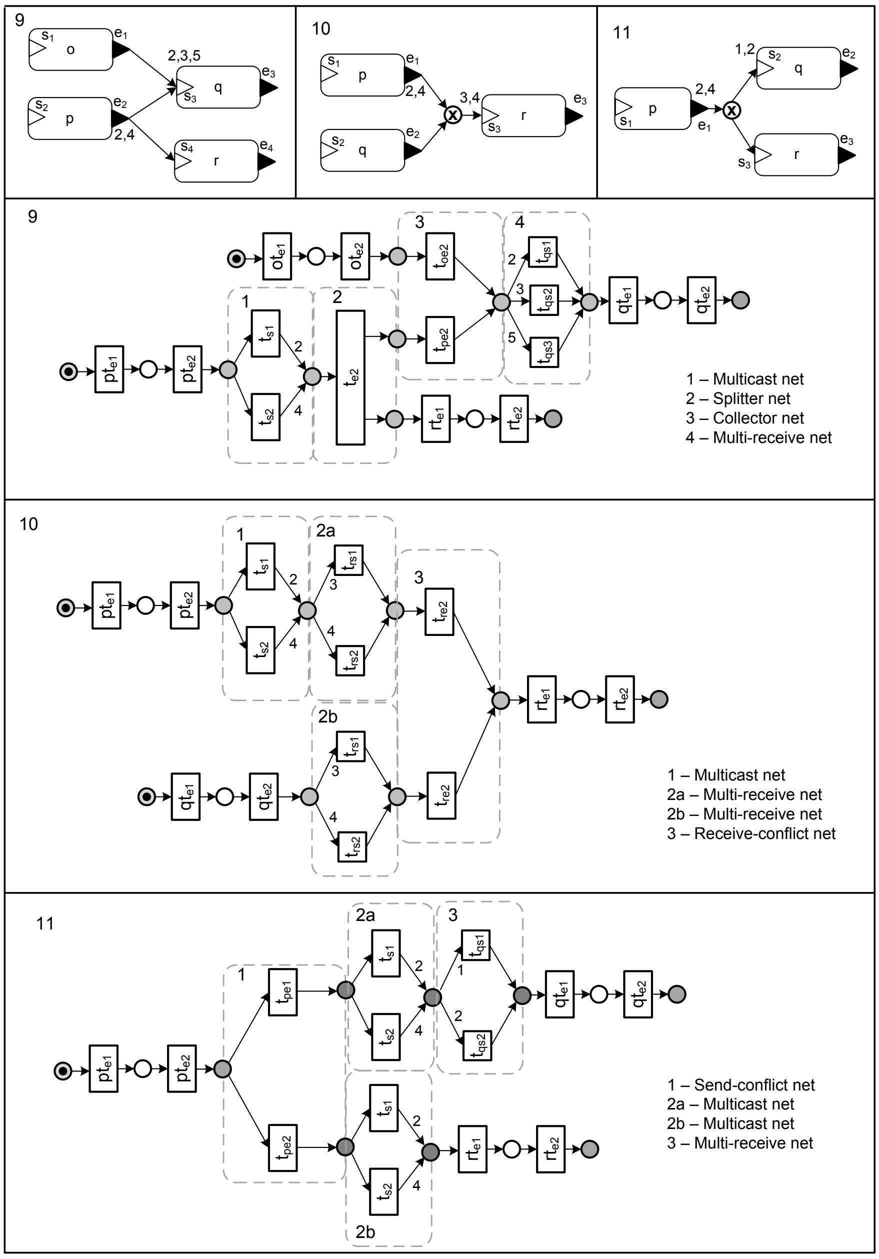


Figure 29: Multi-communication composition rules of several intermediary nets

Composition rule 9 exemplifies that the multireceive and multicast nets are always directly connected to the according input respectively output interface place of the according ONs, unless their original events are part of a conflict relation. Additional splitter and collector nets are then connected to the intermediary multicast respectively multireceive net. The insertion of up to four intermediary nets may be required to connect two processes' ONs. An ON of processes emitting triggers or information flows is first connected to an intermediary multicast net (1) that subsequently is connected to the intermediary splitter net (2) as depicted in Figure 29. The semantics become apparent immediately, both processes receive the same amount of triggers , e. g., two or four as depicted in the multicast net in Figure 29. A similar situation is depicted for processes with receiving events. The multireceive net (4) is directly connected to the input interface place of the according ON which is subsequently composed with a collector net (3). This depicts that the receiving process needs a specific amount of triggers or information flows as input but it does not matter from which process it receives them. The insertion of intermediary nets between sending and receiving processes follows the rule multicast net (1) before splitter net (2), collector-net (3) before multireceive net (4) as illustrated in Figure 29.

Composition rule 10 and 11 describe the composition rules for events that have a non-trivial multiplicity set, a non-singleton postset, and additionally are part of flows that are in conflict. The conflict relation forms a special case and requires the re-ordering of intermediary nets. If information flows or triggers are in conflict with each other the composition of intermediary nets is the other way round than in composition rule nine. The *send-conflict net* and the *receive-conflict net* are fused directly to the output respectively input interface place of the according ONs. The multireceive and the multicast nets are then copied and fused to each flow option represented in the send-conflict net or the receive-conflict net representing an exclusive multi-communication behavior.

Composition rule 10 shows two sending processes that are in conflict. e_1 and s_3 have a non-trivial multiplicity requiring the insertion of intermediary multicast and multireceive nets. As $((e_1, s_3), (e_2, s_3)) \in \chi$ we need to assert that always only one of the two processes, p or q, triggers one instance of r. This is done by connecting the receive-conflict net directly to the input interface place of the receiving ON and attaching the copies of the multireceive net in between the sender and the receiver ON. E. g., in Figure 29, taking the point of view of the receiving process, the intermediary receive-conflict net (3) is directly connected to the receiving ON_r. Two copies of the multireceive net (2a, 2b) which depict the receiving events non trivial multiplicity are composed with the multicast net (1) and the output interface place of ON_q. The receive-conflict compositions follows the rule multicast net

(1) and multireceive net (2a, 2b) before receive conflict net between sending and receiving process. This composition order assures that always only one of the emitting processes triggers or messages a process and that their triggers or information flows do not get mixed up.

Composition rule 11 describes the order of intermediary nets for triggers or information flows where the receiving processes are in conflict. This means that only one should receive the triggers or information flows defined by the multiplicity of the emitting event. This is assured by attaching the intermediary send-conflict net directly to the output place of the emitting ON and afterwards connecting it to copies of the multicast net. The intermediary send-conflict net depicts the choice that determines which process receives the trigger or information flow. The multicast net produces the amount of tokens specified by the multiplicity of the according event as for example depicted in Figure 29. The send-conflict follows the rule (splitter net), send-conflict net, multicast net, multireceive net. The splitter net is only inserted if there is a trigger to a third process, that is always triggered no matter of the choice of the send-conflict net.

In cases of complex composition multiple intermediary nets are necessary, which per default would have interface places with identical names, thus making the nets non-composable. But since such situations can be derived from the relations, the problem can be circumvented by renaming those places as follows.

Definition 42 (Place renaming for intermediary nets)

a) ***Multicast net for*** $(e, e') \in L \cup I$

$$P^I = \{p_e\} \qquad P^O = \begin{cases} \{p_{e'}\} & \textit{if } e\bullet = \{e'\} \wedge \bullet e' = \{e\} \wedge \mu(e') = \{1\} \\ \{p_{e,e'}\} & \textit{if } e\bullet = \{e'\} \wedge \bullet e' = \{e\} \wedge \mu(e') \neq \{1\} \\ \{p''_e\} & \textit{if } |e\bullet| > 1 \vee |\bullet e'| > 1 \textit{ for } e' \in e\bullet \end{cases}$$

b) ***Multireceive net for*** $(e, e') \in L \cup I$

$$P^I = \begin{cases} \{p_e\} & \textit{if } e\bullet = \{e'\} \wedge \bullet e' = \{e\} \wedge \mu(e) = \{1\} \\ \{p_{e,e'}\} & \textit{if } e\bullet = \{e'\} \wedge \bullet e' = \{e\} \wedge \mu(e) \neq \{1\} \\ \{p''_{e'}\} & \textit{if } |\bullet e'| > 1 \vee |e\bullet| > 1 \textit{ for } e \in \bullet e' \end{cases} \qquad P^O = \{p_{e'}\}$$

c) ***Splitter net for*** e *(Let* $e' \in e\bullet$*)*

$$\begin{aligned} P^O =& \{p_{e'} \mid |\bullet e'| = 1 \wedge \mu(e') = \{1\}\} \\ &\cup \{p''_{e'} \mid |\bullet e'| = 1 \wedge \mu(e') \neq \{1\}\} \\ &\cup \{p''_{e,e'} \mid |\bullet e'| > 1\} \end{aligned} \qquad P^I = \begin{cases} \{p_e\} & \textit{if } \mu(e) = \{1\} \\ \{p''_e\} & \textit{otherwise} \end{cases}$$

d) ***Collector net for*** e' *(Let* $e \in \bullet e'$*)*

$$P^I = \{p_e \mid |e\bullet| = 1 \wedge \mu(e) = \{1\}\} \cup \{p''_e \mid |e\bullet| = 1 \wedge \mu(e) \neq \{1\}\} \cup \{p''_{e,e'} \mid |e\bullet| > 1\} \qquad P^O = \begin{cases} \{p_{e'}\} & \textit{if } \mu(e') = \{1\} \\ \{p''_{e'}\} & \textit{otherwise} \end{cases}$$

e) ***Receive-conflict net for*** e' *(Let* $e \in \bullet e'$*)*

$$P^I = \{p_e \mid |e\bullet| = 1 \wedge \mu(e) = \{1\}\} \cup \{p''_e \mid |e\bullet| = 1 \wedge \mu(e) \neq \{1\}\} \cup \{p''_{e,e'} \mid |e\bullet| > 1\} \qquad P^O = \begin{cases} \{p_{e'}\} & \textit{if } \mu(e') = \{1\} \\ \{p''_{e'}\} & \textit{otherwise} \end{cases}$$

f) ***Send-conflict net for*** e *(Let* $e' \in e\bullet$*)*

$$P^O = \{p_{e'} \mid |\bullet e'| = 1 \wedge \mu(e') = \{1\}\} \cup \{p''_{e'} \mid |\bullet e'| = 1 \wedge \mu(e') \neq \{1\}\} \cup \{p''_{e,e'} \mid |\bullet e'| > 1\} \qquad P^I = \{p_e\}$$

◇

Multicast and multireceive nets as well as splitter (send-conflict) and collector (receive-conflict) nets are mirror images of each other, with the roles of pre- and postset, input and output switched. Consider the naming of multicast (multireceive) nets for $(e, e') \in L \cup I$. The name of the only output (input) place depends on whether e (e') has a non-singleton postset (preset) and thus requires a splitter (collector) net [case a_3 above] and whether e' (e) has a trivial [case a_1] or non-trivial [case a_2] multiplicity set.

Splitter (collector) nets have several output (input) places, whose names again depend on the multiplicity of the preset (postset) of the related events. Essentially we distinguish three cases when creating splitter (collector) nets corresponding to the three subsets of the union in the definition of P^O (P^I): 1) no multireceive (multicast) net present, 2) multireceive (multicast) exists and 3) collector (splitter) net present. For the names of input (output) places we need only to differentiate whether a multicast (multireceive) net is present or not. The application of this naming algorithm is illustrated in the use case in Figure 35.

5.4 SUMMARY

A transformation into a Petri net formalism is a common way to provide clear semantics for process modeling languages [80]. Lohmann et al. [80] investigate the transformation of process modeling languages into Petri nets. Those transformations are mainly used for the formalization of the behavioral semantics and the structural and behavioral verification of the source languages [80]. [148, 113, 32, 80] formalize EPCs, BPMN, and other workflow notations with Petri nets and provide them with clear behavioral semantics that can be analyzed. [148, 32, 80, 98] show the effectiveness of formalizing business process models in Petri nets. For instance, Dijkman et al. [32] provide a BPMN to Petri net transformation to clarify ambiguities in the BPMN specification finding a number of deficiencies of the source language.

Similarly, we introduced a transformation of BPAs into ONs to provide clear behavioral semantics of BPAs and enable the analysis of our BPAs (see Chapter 6). Using ONs has many advantages as abundant Petri net verification techniques exist [147]. Our transformation depicts not only information flows but also highlights the trigger relations prevalent in BPAs. While multi-communication has not been the focus of current approaches, our BPA approach allows modeling the sending (receiving) of messages or triggers to many receivers (from many senders) and the sending (receiving) of everal messages or triggers to one receiver (from one sender). As ONs provide no direct means to depict multi-communication, we presented different intermediary net schema that replicate such complex behavior in ONs by combining them accordingly.

After having introduced the transformation of BPAs into ONs in this chapter, we present techniques for their structural and behavioral analysis in Chapter 6. The behavioral analysis builds on the ON transformation of BPAs.

6 BUSINESS PROCESS ARCHITECTURE ANALYSIS

This chapter is based on the published papers [40, 38, 45, 44]. It introduces behavioral and structural BPA analysis techniques. The analysis of BPAs plays an important role for the re-/design of process models and the quality of process model collection. This chapter refers to the BPA properties in Section 4.6 and the BPA to Open net transformation presented in Chapter 5. It contributes analysis techniques to our overall BPA methodology that we introduce in Chapter 7.

Considering the complex interdependencies between processes in organizations depicted in BPAs it is important to find faults and problems to improve and harmonize operations within a company. These unwanted aspects can be of structural and behavioral nature. We introduce different techniques for analyzing the interdependencies of business processes in a BPA in regard to their structural composition and behavioral properties. This in particular is important to detect undesired structural and behavioral aspects in a BPA and to improve them in a second step.

In Section 6.1, we present normal BPA and BPA anti-patterns to examine BPAs for undesired structural interdependencies and detect incorrect behavior. However, those patterns are restricted to only examine the direct interdependencies between only two processes at a time. To overcome such limitations and extend our analysis capabilities, we introduce techniques for the examination of behavioral properties based on an ON transformation of BPAs in Section 6.2. This allows for thorough analysis of BPAs in regard to the interplay of the interdependent processes and BPA correctness criteria.

6.1 STRUCTURAL BPA ANALYSIS

The structural patterns in the following sections describe information flows and trigger relations between processes in synchronous and asynchronous environments. The patterns consist of basic constructs, composite constructs, and constructs of multiple instances.

Normal patterns that depict desired interdependencies between processes are patterns 1-14 depicted in Figure 30 whereas patterns 15-27 are anti-patterns illustrated in Figure 31 that show interdependencies between processes that lead to faulty behavior. The main attention will be given to anti-patterns as they expose erroneous interdependencies.

We differentiate between asynchronous and synchronous environments for those patterns and anti-patterns. Asynchronous environments in which message can be buffered or do not have to be instantly processed are more common in BPM practice. In synchronous environments timeliness and availability matters. Synchronous environments have stronger restrictions on the availability of processes during communication as we assume processes to be able to receive information flows only when being instantiated. In case a process is not instantiated the information will get lost as the receiver does not exist yet and hence is not ready to receive the according information. In asynchronous environments information flows are buffered and are available when required by the process.

The patterns and anti-patterns are depicted in Figure 30 and in Figure 31 in which process p generally is the starting point of reading the patterns. The reading direction of the patterns is from left to right and from top to bottom.

6.1.1 *Basic Patterns*

The very basic interaction patterns with only one relation between two processes are trigger patterns and information flow patterns. One process is linked to another process by a relation between a sending (end, intermediate throwing) and a receiving event (start, intermediate catching). In trigger patterns one process instantiates another process. Information flow patterns depict an information flow that can be observed between two processes after they have been instantiated. These basic patterns are shown in patterns 1-4 in Figure 30.

Trigger Patterns Pattern 1 depicts a process p that triggers process r. Process p finishes with end event e_1 and process r starts with start event s_1. Similarly, pattern 2 shows process p triggering process r but through its intermediate throwing event t_1. Both patterns hold for asynchronous and synchronous environments.

Information Flow Pattern Patterns 3-4 show an information flow from process p to process r that is emitted by a throwing intermediate event or an end event respectively and received by an intermediate catching event. In both patterns process r must be instantiated for the information exchange to occur in synchronous environments. Else the information flow will be lost. In asynchronous environments, r does not need to be instantiated at the moment when process p passes the information flow. The message will be read after instantiation when the process execution reaches the according state of the process where the input is required.

6.1.2 *Composite Patterns*

Patterns 5-8 in Figure 30 show regular composite patterns that display a combination of trigger and information flow relations between two processes p and r.

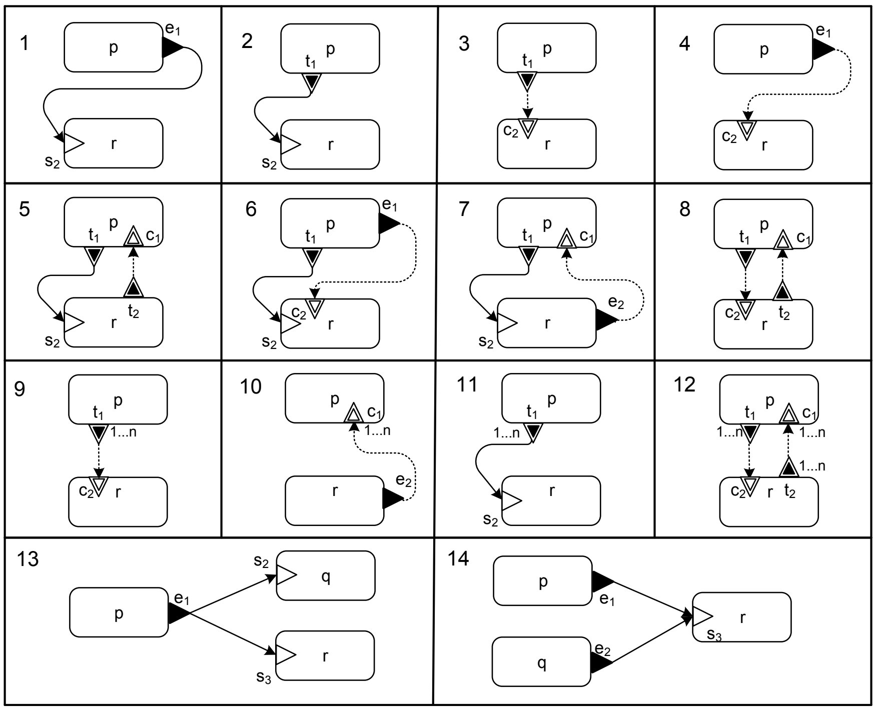


Figure 30: Basic patterns

Information Flow Feedback Patterns Patterns 5 and 7 describe the feedback of information from process r to process p after having been triggered and instantiated by process p. In pattern 5 and 7 process r is triggered by a throwing intermediate event. During its execution process r sends an information flow to process p through a throwing intermediate event in pattern 5 and through an end event in pattern 7. These pattern are similar to general request-reply patterns. Pattern 7 can be used to identify and represent sub-processes.

Unidirectional Interaction Patterns In contrast to patterns 5 and 7, pattern 6 displays only unidirectional interaction from process p to process r. Process p instantiates process r and sends a message to process r when it finishes. In synchronous and asynchronous environments process r has to wait for process p to finish before process r can continue its execution.

Send-Receive Pattern In pattern 8, process p passes an information flow to process r through an intermediate throwing event and vice versa. Both processes need to be active in synchronous and asynchronous environments. However, process r could be instantiated at later time in asynchronous environments so that process p has to wait for its response and vice versa.

Broadcast In pattern 13 a broadcast of the end event of process p to processes q and r can be observed. Both processes will be triggered and instantiated. This pattern describes one type of multi-communication.

Collector In pattern 14 process r can either be triggered by the end event of process p or by the end event of process q or by both. If both processes send a trigger then process r is instantiated two times. Patterns 13 and 14 include the according conflict relations.

6.1.3 *Multi-Instance Patterns*

To describe multi-instantiation patterns we introduced multiplicities in Definition 29 that define the number of triggers and information flows sent or received in a relation between two processes. The three vertical lines depicted in process r in patterns 9-12 of Figure 30 represent the possible parallel execution of multiple instances of process r. Note, for the multi-instance pattern, the multiplicity of the events have a different interpretation if the communication between many instances is considered. E. g., in pattern 9 the multiplicity of the intermediate throwing event distributes information flows to each instance of process r. Each instance of process r receives one information flow and not one instance receives all information flows sent from p. This is represented by the vertical bars on the process icon. If also the receiving event has different multiplicities then each process instance can receive a different amount of information flows as specified its event's multiplicity set for that run. Multi-instance patterns depict resource patterns that are important to consider, e. g., when assigning roles to employees and processes.

One-to-many-Broadcast Pattern 9 displays a one-to-many broadcast pattern where process p sends information flows to many instances of process r. In synchronous environments as many instances of r need to be active as information objects are sent by p. In asynchronous environments the instances could also become active in sequential order.

Information Sink Pattern 10 displays an information sink. It extends pattern 4 with multiplicities. Process p collects many information objects from several instances of process r that run in parallel. In synchronous environments all process instances need to be active. In asynchronous environments p can become active at a later stage to receive all information objects from process r.

Trigger Fleet Pattern 11 depicts similar behavior like pattern 2. However, process p triggers many instances of process r. In synchronous environments all process instances of process r become active at the same time. In asynchronous environment, the instances of r could become active at different points in time.

Pattern 12 is a combination of pattern 9 and 10 whereas the process instances of process r also send many information objects to process p. Process p collects as many replies as needed and can then continue with its execution.

6.1.4 *Basic Anti-Patterns*

There are four basic process anti-patterns that demonstrate self-reflexive/looping behavior and expose incorrect structures of a BPA. They are depicted in patterns 15-18 of Figure 31. Anti-patterns that are considered regular patterns in asynchronous environments are marked with an asterisk "*" in Figure 31 and in the heading of the pattern description.

Loop Patterns 15 and 16 describe looping behavior in which a process p triggers itself either through its own throwing intermediate event or through its own end event. In pattern 15 process p triggers itself upon finishing. This relation should not occur in a BPA as such process could never be instantiated, unless zero is an element of the start event's multiplicity set. Also if instantiated once, process p would trigger itself infinite many times as there is neither another end event, nor is there a zero in the end event's multiplicity set, nor another trigger or information flow that is in a conflict relation with the self trigger relation. Similarly in pattern 16, process p triggers itself with its throwing intermediate event t_1. It can never become active as it waits for its own throwing intermediate event to trigger itself. Also if instantiated once, infinite many instances of process p would be created. The end event in pattern 15 and the intermediate throwing event in pattern 16 respectively do not appear in another information flow or trigger relation that is in conflict with the self trigger flow. Such a conflict or the inclusion of zero in the end event's multiplicity set would provide a way out of the dead process or livelock observed. Such looping behavior that shows dead processes or leads to livelocks is undesired in synchronous as well as asynchronous environments.

Dead Event Pattern 17 of Figure 31 depicts a dead event. When process p is active it waits for an information flow from its own end event. t_1 will never occur so that process p will never end. c_1 is a dead event as it never occurs. Dead events hint at dead processes and are an indicator for violating BPA correctness criteria.

Self Messaging Pattern 18 does not formally depict undesired behavior. However, a process p that passes an information object to itself, displays an improper structure and improper behavior. The throwing intermediate event t_1 sends an information object to its catching intermediate event c_1.

If the receiving event was supposed to happen before the sending event, i. e., both intermediate events were switched around, this self messaging would lead to a deadlock. In this case the receiving event would be dead and the following sending event as well. In both synchronous and asynchronous environments this could lead to process p not finishing as the information flow to itself does not start.

6.1.5 *Composite Anti-Patterns*

The anti-patterns in this group consist of lost information flow patterns, inhibiting information flows or triggers, and looping behavior.

Trigger Loop Deriving from patterns 15 and 16, patterns 19, 20, and 25 depict looping behavior involving two processes.

In pattern 19 and 20 process p triggers process r through a throwing intermediate event t_1. Process r in return triggers process p so that the combination of both interdependencies resolves in a dead structure. This symmetric behavior describes a loop spanning over two processes in which neither process can be instantiated as they depend on being triggered by each other. Also, if once active, multiple instances of process p could exist in both patterns at the same time. In pattern 19 multiple instances of r could possibly run in parallel. The pattern structure could never terminate as the processes instantiate a process instance of each other over and over again. Pattern 25 also depicts looping behavior, except that process p when finishing triggers process r. Either pattern cannot be instantiated as both processes are triggered by each other.

Inhibitor and Lost Information Flows Patterns 21-24 in Figure 31 show different constructs in which information flows never happen, inhibit process execution, or would be lost in synchronous environments. In pattern 21, process p is triggered by process r and is supposed to send an information flow to r. Process r is inhibited by the throwing event t_1 of process p which cannot occur as process p is triggered by the intermediate event t_2 of process r. This leads to process p inhibiting the execution of process r.

An information flow gets lost when the receiver process cannot be active while the information flow is sent. In patterns 22 and 24 process p triggers process r when it finishes. However, process p cannot terminate and deadlocks as it waits for the information flow of process r. The catching intermediate event and the end event of process p are dead. Consequently, process r is dead and will never send an information flow to process p. If process p would receive an information flow from another process, the information flow sent by process r would still be lost as structurally process p finishes before process r starts. These patterns do not only depict lost flows but also dead events and consequently deadlocks and dead processes.

Unidirectional Information Flow Loss Pattern 23*, 26*-27* describe a unidirectional relation information flow loss in synchronous environments. Process p is active and wants to send an information flow to process r. However, process r is not instantiated yet, as it is triggered by either the sending intermediate event (anti-pattern 23*, 26*) or the end event (anti-pattern 27*) of process p. Hence the information flow will be lost in synchronous environments as process r cannot be instantiated and receive an information flow through its catching in-

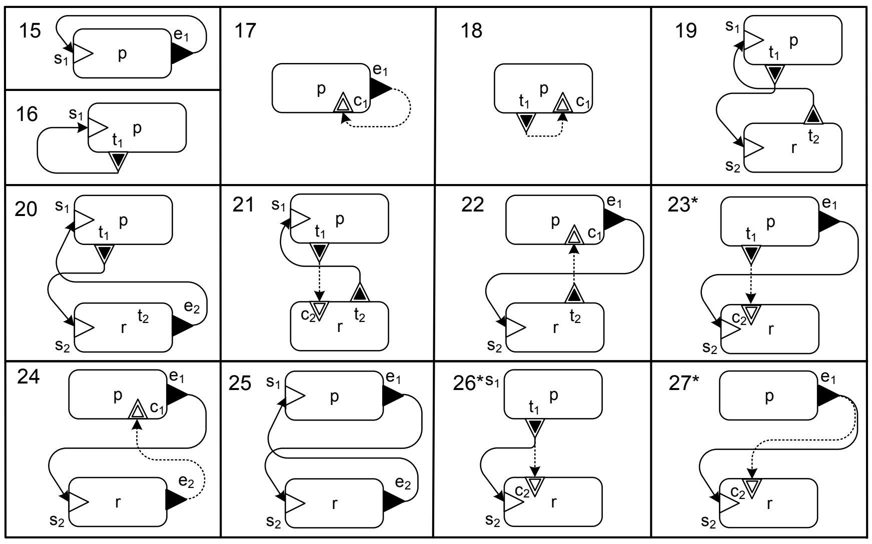


Figure 31: BPA Anti-patterns

termediate event at the same time. In asynchronous environments patterns 23, 26, and 27 describe regular behavior as the message object will be buffered until the according process will be active and able to process it.

6.1.6 *Application of Pattern and Anti-Patterns Analysis*

We evaluate our pattern based analysis technique by applying it to a subset of business process models from the SAP Reference Model collection [21]. For this we examined the SAP Reference Model collection for interdependencies between business processes and constructed BPAs according to the interdependencies found.

Constructing a business process architecture for the SAP reference model is complicated by three factors. First, the SAP Reference Model, has a control flow semantics that is difficult to interpret automatically, because it is known to contain errors in the control flow and EPCs, the used modeling notation in particular, lacks clear semantics [99, 148]. The extraction algorithms proposed in Section 7.2 cannot be applied to the SAP Reference model because most of the underlying process models in the SAP Reference Model are not structurally sound and not block structured. Hence, the manual construction of BPAs from the SAP Reference model and a manual investigation of the collection was inevitable. [25] shed light to the difficult instantiation semantics of EPCs looking at process models in the SAP reference model with many start events. Many process models of the SAP process collection have several start events that are connected with XOR, OR, or/and AND gateways. On the process model level it is not always fully clear how to interpret such complex semantics due to under-specification of EPCs for example. Abstracting from such highly complex semantics to BPA level is difficult and restricted by the quality of the process model semantics on process model level which becomes obvious on BPA level. As most of the process models in the SAP Reference Model have several start and end events we loosened the restriction of our BPA definition in Definition 29, such that a BPA process can have several start and several end events. This allows us constructing BPAs from highly complex process models and process model interdependencies but impacts the interpretation of the BPA semantics and the pattern based analysis as well.

Second, the SAP Reference Model does not explicitly distinguish between start, end and intermediate events, while a BPA distinguishes between them. To identify the events, we used the following definitions. A start event is an event that is meant to occur before any activity. An end event is an event that is meant to occur after all activities have occurred. An intermediate event is an event that is meant to occur after an activity has already occurred and before all activities have occurred. Of course, these assumptions leave room for interpretations but are an appropriate solution to the problem posed by lack of semantics of the modeling notation. To determine whether an event is expected to occur at all for a single instance of a process, we assume that the processes are meant to be one-safe in a token-based execution semantics. For example, if a process has two start events A and B that are followed by an XOR-join, then we assume that either A or B should occur in a single process instance, but not both.

Third, the SAP Reference Model also does not distinguish explicitly between trigger and information flow relations or between synchronous and asynchronous relations. Instead, it distinguishes between conditions that can become true. We interpret each of these conditions as a synchronous trigger relation, as a condition that becomes true, becomes true at all places in the architecture at the same time.

Using these assumptions, we constructed BPAs for the various branches, like the one shown in Figure 32, and subsequently we identified the architectural patterns that are shown in Table 1. We investigated **7** branches of the model with a total of **12** subbranches. These branches contain **95** of the total collection of 604 models. The interaction observed between process models remained within their branches. However, interaction across subbranches could be observed quite frequently. Figure 32 shows a process architecture for one of the smaller branches of the SAP Reference Model, consisting of only leaf processes. The architecture provides some interesting insights. For example, it is visible that process c can be executed multiple times and process b collects the results of the various instances of process c as process b monitors process c. Process d appears to be highly relevant as it has many relations. It can be triggered by four different processes.

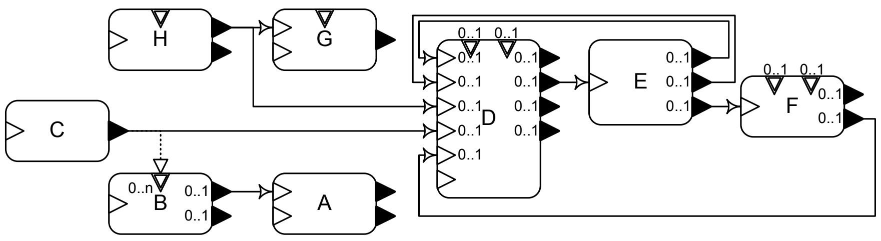


Figure 32: Part of the SAP reference model process architecture.

We can observe multi-communication pattern 13 two times; process h triggering processes g and d, and process c triggering process d and sending an information flow to process b.

The BPA shown in Figure 32 is not well-formed due to processes h, g, f having unconnected catching intermediate events. Processes h and g deadlock and cannot terminate as the source of their required information flow is not specified. Similarly, processes d and f may deadlock but their information flow input is optional, such that there can be terminating runs. The BPA contains two loops (anti-pattern 25) spanning over up to two processes that indicate possible livelocks and a possible violation of our BPA correctness criteria. As these processes have also other end events, we cannot make a clear statement except from indicating problem areas in the BPA. It exhibits one loop along three processes that cannot be detected by our anti-pattern approach as we analyze only direct relations between two processes.

During our examination of 95 business processes of the SAP Reference Model we mainly found normal patterns that derive from patterns 1-4. In general we found more trigger relations than information flow relations. Trigger relations with end events triggering a start event occurred more often than trigger relations that were initiated by intermediate throwing events. We observed more information flow relations between to intermediate events than information flow relations between end events and catching intermediate events. Two of the analyzed branches showed a correct structure as they did not expose any anti-patterns.

Table 1: Patterns in SAP-Reference model

Pattern	Counts	Pattern	Counts
Trigger (Pattern 1 and 2)	95	Self-Messaging	2
Flow (Pattern 3 and 4)	13	Anti-Pattern 25	3
Broadcast	10	Anti-Pattern 20	3
Loop (Anti-Pattern 15 and 16)	5	Anti-Pattern 23	2

In the other five branches **13** occurrences of anti-patterns could be observed. Anti-pattern 23 was found two times and loop anti-patterns 15 and 16 in total five times. However, one occurrence of the loop anti-pattern and the two counts of anti-pattern 23 could be considered regular behavior when taking their hidden workflows into consideration as each process exposes several start and end events. Anti-pattern 20 could be observed three times in derivative form and anti-pattern 25 three times.

This evaluation shows that we can apply our pattern based technique to large and complex sets of process models. It indicates structural errors and hence supports the improvement of the quality of a process model collection in regard to consistent process model interdependencies. Despite that, our pattern based analysis technique is limited to the analysis of the direct relations between two processes only. Erroneous behavior that is caused by indirect interdependencies cannot be detected. An ON based analysis technique presented in the next sections overcomes this limitation and supports the analysis of BPA correctness.

6.2 BEHAVIORAL BPA ANALYSIS

In Chapter 5, we introduced a BPA transformation to ONs to provide clear semantics for the behavior of BPAs and clarify the behavior of BPAs with multiplicities and multi-communication. Besides providing clear behavioral semantics, Petri net based formalisms are commonly used for modeling and analyzing distributed, concurrent, and asynchronous systems [147, 148, 113, 32, 80, 98]. We will use the ON transformation approach for examining our patterns for their behavioral properties. Besides the analysis for BPA correctness criteria defined in Definition 32 and Definition 33, we also examine the pattern and anti-patterns for known behavioral workflow properties and BPA correctness criteria. In a later step, we highlight the analysis of multi-communication by examining BPAs with multiplicities in regard to the BPA correctness criteria defined in a larger use case.

6.2.1 *General Analysis Procedure*

To analyze the correctness of BPAs (BPA subsets) we transform and compose them into one ON according to the transformation algorithm presented in Section 5.2. At the same time we create CTL-formulae and state predicates that express the BPA correctness criteria for the resulting composed ON. For each ON representation of a BPA the CTL-formulae and state predicates that express the correctness criteria are created individually. For each criterion of the BPA correctness criteria a verification task for the model checker is created. Both results of the transformation, the ON and the BPA correctness criteria in form of CTL-formulae and state predicates are used for model checking to determine if the BPA correctness properties can be satisfied by the composed ON representation of the BPA. If the transformed ON successfully passes all the verification tasks the BPA is correct.

A terminating BPA run is characterized by the final place or places of the net being marked with one or more tokens and any other place in the net being unmarked. If such a state is reachable in the state space, the BPA has a terminating run and thus is correct.

Lazy termination of a BPA run can be concluded if there is a path in the state space leading to a final marking, such that each process terminates at least once, if it is instantiated at all in that run. In lazy terminating BPA runs not terminated instances and pending messages are allowed, as long as for each instantiated process at least one instance terminates and the final marking is reached. This property has similarities with relaxed soundness of workflow nets and weak soundness in the context of process choreographies.

A lazy terminating BPA run in most cases contains lost triggers or information flows. A lost trigger or information flow gets emitted to an event that may never become ready or may not become ready again. This is depicted in the state space when the final marking is reached but there are still remaining tokens in the ON. E. g., a transition representing a catching intermediate event has a token on its interface input place but the according process is not instantiated so that the transition cannot fire. Lost trigger and information flow also often appear with multi-communication if the amount of instances and information flow sent or multiplicities of catching and throwing event do not match.

If a final marking is not reachable, the BPA contains only deadlocks and infinite BPA runs (livelocks). Those can automatically be detected by LoLA[1] [132]. Dead processes are found by searching the state space for all those initial places, that are never marked. Similarly, we identify dead events by searching the state space for all those input places of transitions that represent BPA events that are always unmarked.

To examine whether a process terminates, the state space is searched for a path on which the initial place of a process representing its start event has been marked once and eventually the end places of the process representing its end event has been marked once as well. Additionally, the final marking of the BPA must be reachable on this path for a BPA to be correct. Else the process can terminate but does not participate in a (lazy) terminating run.

In our implementation of the BPA analysis, we used the LoLA built-in verification tasks for model checking and applied them to the com-

1 Low Level Petri net Analyzer, http://service-technology.org/lola

posed ON. The implementation of our BPA analysis tool is described in Section 9.2.

6.2.2 *Analysis of Patterns and Anti-Patterns*

In previous section we proposed 27 BPA patterns and anti-patterns to identify desired and undesired structural and behavioral properties. These patterns provide a first means to detect errors between two processes in a BPA. However, for large BPAs their application becomes complex. In order to assess their properties we examine the BPA patterns by applying our BPA to ON transformation and categorize them.

In total we examine all 27 patterns, of which 14 are considered regular patterns and 13 anti-patterns respectively. We analyze those patterns in regard to the BPA correctness criteria defined in Definition 32. We use common workflow correctness criteria like soundness where applicable to support our analysis. As BPAs do not necessarily comply to the workflow net or workflow module structure and encompass a larger context, workflow net and workflow module correctness criteria cannot be applied in all cases. Due to that, BPA properties in Chapter 5 were designed. In cases where the common workflow net properties are applicable they support the analysis.

A representative set of BPA patterns, depicted in Figure 33, will serve as example to illustrate this approach. Figure 33 shows solely the resulting fused ONs from the transformation of the patterns and anti-patterns. The usual dotted demarcation lines for ONs are not visualized in Figure 33 for clarity reasons. The patterns shown in Figure 33 were selected as they cover most of the structural aspects and show the various behavioral properties observed in BPA patterns.

Our course of action consisted of three steps, firstly the transformation of BPA patterns to ONs, secondly the analysis of the resulting ONs for their structural and behavioral properties with LoLA, and thirdly the categorization of transformed BPA patterns. The transformation of BPA processes and their trigger and information flow relations results in ONs with different number of start and end places.

The initial approach presented in [38] included a transformation into WF nets for structural unsound nets which we do not apply here. Using ONs for the analysis does not require this step anymore and allows for direct analysis of the resulting ON.

The examination of the BPA patterns shows interesting results. The results can be grouped into three categories depicted in Table 2, Table 3, and Table 4. The patterns in the tables were sorted along their structural properties, i. e., the number of start and end places, and the resulting behavioral properties, e. g., if they are correct, dead or live. After the first step of our analysis, the transformation of BPA patterns into ONs, most of the BPA patterns are structurally not sound.

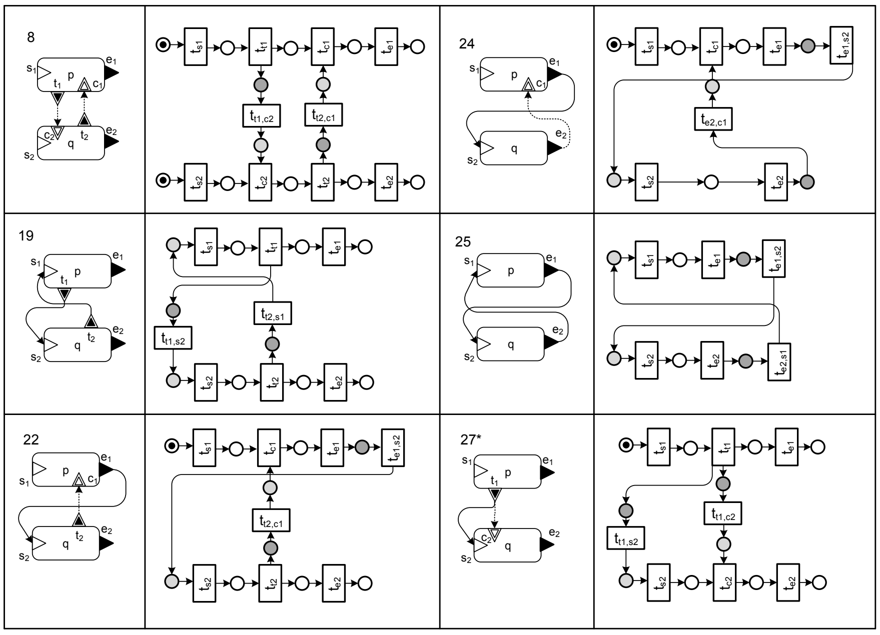


Figure 33: Representative BPA patterns mapped to ONs

Either they have several or no start, and several or no end places, or a combination of both.

The ***Correct ONs*** category includes structurally, behaviorally sound, and bounded ONs. These are the ONs that depict the regular patterns 1, 6, 7, 18, 23, 26, and 27. Patterns 23, 26 and 27 are considered anti-patterns in environments of synchronous communication which does not apply for the asynchronous ON environment. The BPA patterns 2, 3, 4, 5, 8, 9, 13 and 14 have several start and end places but are bounded as well. Those patterns meet all BPA correctness criteria and are *complete terminating*. The multi-instance patterns 10, 11, 12 are only *terminating* as some of their runs have not matching multiplicities so that some runs may be only *lazy terminating* and some runs may even *deadlock*. However, all multi-instance patterns are well-formed and have at least one *terminating* run. The analysis results of the anti-patterns that become correct patterns are shown in Table 2. Using only WF net criteria pattern 26 would be considered incorrect.

The ***Dead Net*** category consists of all patterns that are dead. Dead nets are ONs that can never fire for lacking an initial place, e. g., pattern 19 in Figure 33. They are structurally not sound. The anti-patterns 15, 16, 19, 20, and 25 are grouped into this category. None of these patterns depict a well-formed BPA as they do not have any independent start event ($\mu(e_s) \neq \emptyset \wedge 0 \notin \mu(e_s)$). If those patterns would be started by a second trigger relation or a zero in their multiplicity set they would result in livelocked BPAs but their processes could start and terminate. The revived nets would not have terminating runs either. We call these dead nets that get activated by conceptually

Table 2: Properties of correct BPA patterns

Properties	**Patterns**			
	18	**23**	**27**	**26**
Start Place	1	1	1	1
End Place	1	1	1	2
Struct. Sound	yes	yes	yes	no
WF net Sound	yes	yes	yes	no
Weak Sound	yes	yes	yes	no
Relaxed Sound	yes	yes	yes	no
Bounded	yes	yes	yes	yes
Live	no	no	no	no
Dead Event	0	0	0	0
Deadlock run	no	no	no	no
Livelock run	no	no	no	no
Well-formed BPA	yes	yes	yes	yes
Dead processes	0	0	0	0
Terminating processes	1	2	2	2
Lazy terminating	no	no	no	no
BPA terminating	yes	yes	yes	yes
Correct BPA	yes	yes	yes	yes
Dead BPA	no	no	no	no
Deadlocked BPA	no	no	no	no

inserting an external trigger, revived nets. In the ON presentation the revived dead nets would produce an unlimited amount of tokens. The properties of those patterns are presented in Table 3.

The ***Deadlocked BPA*** category consists of patterns of which all runs deadlock. Anti-patterns 17, 21, 22, and 24 form this category. All of the patterns however are bounded. Pattern 17 and 24 are not well-formed. All patterns depict incorrect BPAs as they have dead events and either one or both processes do not terminate. Three of the patterns also have dead processes that cannot be instantiated in any run. Only pattern 22 is structurally sound but also deadlocks. Table 4 shows the properties of those patterns.

As all regular patterns have the same properties they are considered part of the correct pattern group but not listed in the table. The analysis showed that most of the BPA patterns and anti-patterns are structurally unsound. Eight anti-patterns are not well-formed, those include all patterns from the dead net pattern category and dead-

Table 3: Dead BPA patterns and their properties

Properties	**Patterns**				
	15	**25**	**16**	**20**	**19**
Start Place	0	0	0	0	0
End Place	0	0	1	1	2
Struct. Sound	no	no	no	no	no
WF net Sound	no	no	no	no	no
Weak Sound	no	no	no	no	no
Relaxed Sound	no	no	no	no	no
Bounded	yes	yes	no	no	no
Live	no	no	no	no	no
Dead Event	2	4	3	5	6
Deadlock run	no	no	no	no	no
Livelock run	no	no	no	no	no
Well-formed BPA	no	no	no	no	no
Dead processes	1	2	1	2	2
Terminating processes	0	0	0	0	0
Lazy terminating	no	no	no	no	no
BPA terminating	no	no	no	no	no
Revived terminating processes	1	2	1	2	2
Revived Livelocked run	yes	yes	yes	yes	yes
Revived BPA terminating	no	no	no	no	no
Revived BPA lazy terminating	no	no	no	no	no
Correct BPA	no	no	no	no	no
Dead BPA	yes	yes	yes	yes	yes
Deadlocked BPA	no	no	no	no	no
Revived Livelocked BPA	yes	yes	yes	yes	yes

lock BPA patterns. This shows that structural malformedness is an indicator for behavioral malfunctioning. Structural well-formed BPA patterns, however, do not necessarily lead to correct behavior as seen for patterns 21 and 22.

The regular BPA patterns represent desired behavior and result in correct BPAs. Patterns 1, 6, and 7 are sound and result in correct BPAs. The BPAs are complete terminating as all runs terminate. Soundness is a sufficient but not a necessary criteria to show BPA correctness.

Table 4: Properties of deadlock BPA patterns

Properties	**Patterns**			
	22	**17**	**24**	**21**
Start Place	1	1	1	1
End Place	1	0	0	2
Struct. Sound	yes	no	no	no
WF net Sound	no	no	no	no
Weak Sound	no	no	no	no
Relaxed Sound	no	no	no	no
Bounded	yes	yes	yes	yes
Live	no	no	no	no
Dead Event	5	2	4	6
Deadlock run	yes	yes	yes	yes
Livelock run	no	no	no	no
Well-formed BPA	yes	no	no	yes
Dead processes	1	0	1	1
Terminating processes	0	0	0	0
Lazy terminating	no	no	no	no
BPA terminating	no	no	no	no
Correct BPA	no	no	no	no
Dead BPA	no	no	no	no
Deadlocked BPA	yes	yes	yes	yes

In many cases the soundness criteria cannot be applied as connecting many processes along their interdependencies results in structurally unsound nets as we can see with patterns 3, 8, or 14 for example. Those patterns depict correct BPAs that are well-formed and terminating. These examples show that a BPA does not have to be sound nor structurally sound to be correct as well as the limitations of the soundness criteria for BPA scenarios that involve several processes.

In asynchronous communication environments anti-patterns 23, 26, and 27 depict correct BPAs as the messages and triggers sent can be buffered and do not get lost. Hence, the BPA have terminating runs.

Reviving dead nets by adding another trigger relation from another process or adding zero to the multiplicity set of one start event, results in livelocked BPAs, e. g., pattern 15, 16, 19, 20 or 25. Those patterns already show by their structure that they are dead and not well-formed for not having an independent start or end event. Hence, they must be changed structurally.

Business Process Architectures dead net or livelock patterns could possibly result into correct behavior if their events took part in further information flow or trigger relations which would be in conflict with each other. In this case, the processes would exhibit loops which could be interrupted. Looking at the structure of the resulting ONs we see that the different soundness criteria are not suitable for BPAs that involve many processes due to their structural restrictions. There are many patterns that show correct BPAs but are structurally unsound. This will become even more clear when looking at multiplicities and multi-communication depicted in BPAs. Our analysis of the patterns shows their validity to indicate behavioral problems in BPAs due to their structural composition. However, the pattern analysis is limited to a scope of two processes only.

6.3 ANALYSIS WITH MULTIPLICITIES

To emphasize the analysis of multiplicities and evaluate our approach on BPA subsets larger than the BPA patterns analyzed in previous section, we consider the use case of applying for a restaurant business permit combined with a construction permit. An entrepreneur would like to enlarge the facilities chosen for her restaurant, as well as make changes to existing building structures. Figure 34 shows the BPA consisting of seven business processes and their interrelations. E. g., business process p_1, the restaurant business permit application, triggers three business processes p_2, p_3, p_4; the applications for the restaurant permit, the construction permit, as well as the building conversion permit. It is common in the public sector that some of the public administration processes are executed several times by different roles. The construction permit application process p_4, triggers multiple instances of the expert's report process, as depicted by the multiplicity of its throwing event. Another peculiarity is the catching event of the final construction permit evaluation process, which requires between four and six messages from process p_6. This process is responsible for checking formal requirements of the expert reports. Only then the final decision on the construction permit can be taken.

On first sight the BPA model seems to comply to the BPA correctness criteria, except for the disparity between the numbers of created instances of process p_5 and the expected messages in process p_7.

Figure 35 shows the transformation of the BPA with multiplicities into an ON which is then used to verify correctness. Structurally the BPA is well-formed. It has an independent start and an independent end event, and all other events are part of a trigger or information flow relation. The analysis with LoLA [132] showed that the resulting Open net is structurally sound. However, the ON exhibits no firing

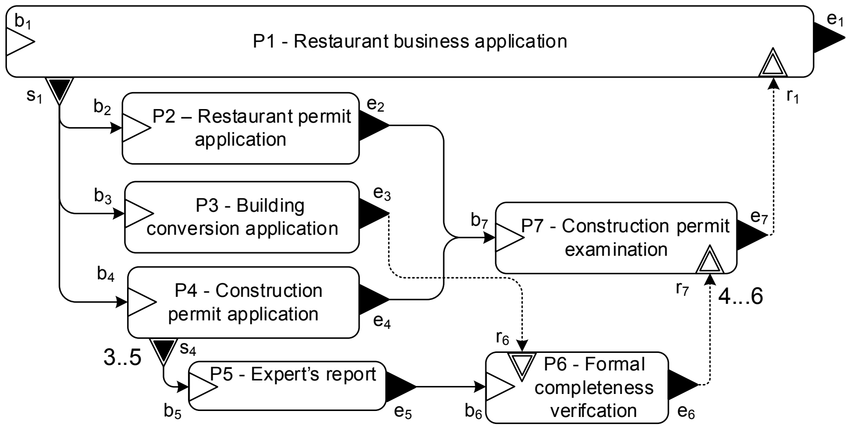

P1 - Restaurant business application
b1
e1
s1
r1
b2
P2 – Restaurant permit application
e2
b3
P3 - Building conversion application
e3
b4
P4 - Construction permit application
e4
b7
P7 - Construction permit examination
e7
r7
4...6
r6
3..5
s4
P5 - Expert's report
b5
e5
b6
P6 - Formal completeness verifcation
e6

Figure 34: Business restaurant permit application

sequence that would mark the place p_{e_1} which corresponds to the end event of the application process and is the desired final marking. This means that all possible BPA runs are neither terminating nor lazy terminating. Instead all runs deadlock, i. e., the ON reaches markings in which no transitions are activated.

All those deadlocks have tokens on p'_{b_7} but do not activate t_{r_7} because no tokens are on p_{r_6} Therefore the catching event r_6 in process p_6 fires only once and is dead afterwards. As a consequence process p_7 is blocked after being initiated twice as not enough incoming message are sent by process p_6. The intermediate catching event r_7, r_1 and the end event e_1, e_7 of process p_1 and process p_7 are dead.

All business processes are instantiated, however neither process p_7 nor process p_1 can terminate and only one instance of process p_6 terminates. Consequently the BPA of our use case is not correct in regard to the BPA correctness criteria. Without considering multiplicities the transformed BPA would have a lazy terminating BPA run as all processes except from one instance of p_7 could terminate. A strong contrast to the problems found considering multiplicities. Considering and analyzing multiplicities in process interaction reveals important aspects for ensuring correct BPAs. Current approaches on process model level like workflow modules [91] or service choreographies [28] do not directly allow the modeling and analysis of process multi-communication, i. e., the examination of the sending and receiving of multiple triggers and information flows as well as the interaction of multiple process instances. As the fist approach on PA level, our BPA approach provides a structural and behavioral analysis that also includes multi-communication constructs.

The BPA analysis based on the ON transformation extends the pattern and anti-pattern based approach with a powerful formal analysis of BPAs. Despite that, it can only be applied for asynchronous communication exchanges in which information flows and triggers are buffered until the process is ready to read them. The use of ONs for

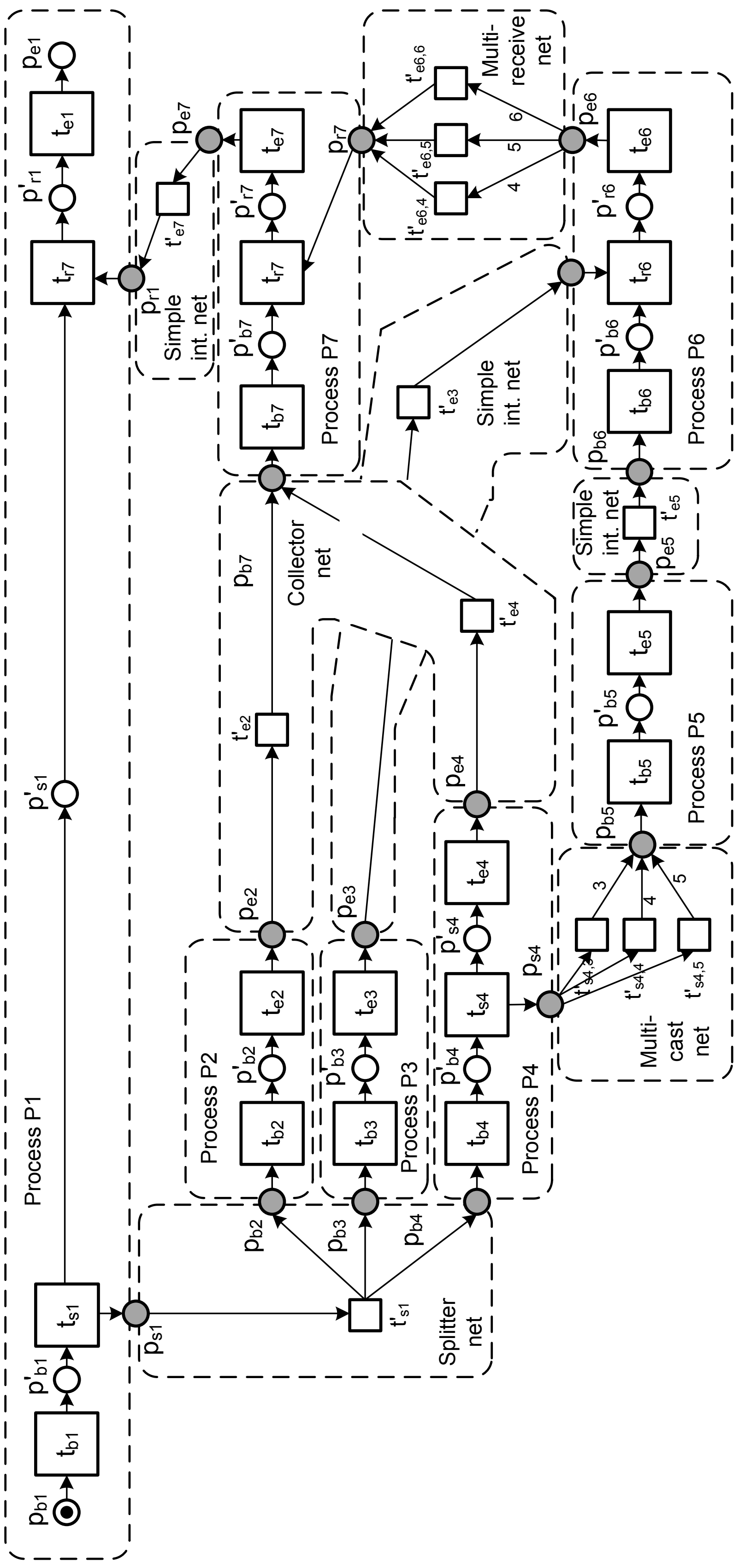


Figure 35: Extended ON BPA transformation

the analysis of BPAs has many advantages as different analysis tools exist. The transformation to ONs and its analysis bears the common disadvantages of state space explosion for BPAs with many business processes, or frequent use of large multiplicity sets and multi-communication. The analysis on BPA level provides means to examine process interdependencies in an early stage of the design phase. When process models already exist, abstracting from a large part of the business process model reduces the state space significantly in contrast of checking interconnected business process models on process model level.

6.4 SUMMARY

In this chapter we introduced structural and behavioral analysis techniques for BPAs. On PA level no comparable techniques exist that allow for the analysis of PAs. Initial attempts on PA analysis are described by Kolliadis et al. [70] and Green and Ould [56] that rather suggest manual analysis based on their approaches. We introduced a pattern based analysis technique and demonstrated it can be used to detect incorrect behavior in BPAs. However, we cannot claim that this approach can be used to uncover all problems related in regard to the BPA correctness criteria stated, e. g., deadlocks or livelocks. Problems that are not captured as an anti-pattern will not be detected. Indicating structural errors and supporting improvement of BPAs, this approach is limited to direct interdependencies between processes.

In regard to the analysis of BPAs, the analysis of multiplicities and multi-communication is of importance which has not been looked at in detail by current approaches. Although ONs have been applied to various use cases on inter-organizational service compatibility, composition and refinement [158, 92, 79, 78], they do not provide direct means to model and analyze multi-communication. We introduced a novel approach that allows to analyze BPAs, i. e., subsets of interdependent processes with their inherent multiplicities and multi-communication. As noted in Section 5.2 we assume one overall case per BPA run, hence we did not consider correlation issues. The introduction of correlation would require the adaption of our ON transformation and analysis approach with a correlation mechanism, e. g., a token identifier and special input and output conditions for transitions. Zhao et al. [175] present correlation Petri nets (CorPn) which may be suited for such purposes. This should be examined in future work. The presented analysis of BPAs is incorporated in our BPA methodology in Chapter 7 which describes the application of our BPA approach top-down and bottom-up.

Part III

EXTENDED CONCEPTS AND APPLICATION

7 BUSINESS PROCESS ARCHITECTURE METHODOLOGY

Parts of this chapter are based on the published paper "From Process Models to Business Process Architectures: Connecting the Layers" [39]. Having introduced the main concepts of BPAs this chapter focuses on the application of BPAs. It presents our BPA methodology and provides some insight on how to use our BPA approach bottom-up and top-down. In this regard it describes a BPA extraction and a BPA to process model decomposition technique.

After having discussed the core concepts of BPAs, their semantics, and behavior, we describe the concrete application of BPAs in an overall methodology. In general there are two ways to apply PAs, top-down or bottom-up. Both come with different preconditions. In the following we discuss the application of our BPA approach in both cases and describe our overall application methodology. Both application ways lack one view, the top-down approach misses business process models in the beginning, whereas the bottom-up approach misses an overarching structure and overview. At first we introduce our BPA methodology in Section 7.1 and the according extraction and decomposition techniques in subsequent sections.

7.1 BUSINESS PROCESS ARCHITECTURE METHODOLOGY

Our BPA approach envisions three conceptual layers for a process model collection repository, the process model level, the level of BPA subsets, and the BPA compendium, a general overview of all BPA subsets. The heart of our framework is our BPA formalism. It allows to define business processes on a very abstract level and their interdependencies to other processes. It can be used to visualize business processes and their interdependencies as well as additional meta information of process models like cost, resources, or risks, and to reason about this information and its context. In Section 6.1 and Section 6.2 we provided two analysis techniques on BPA level that allow for early analysis of interdependencies between business processes.

Based on the BPA foundations, our Business Process Architecture application methodology describes the use of our BPA concept and combines it with additional techniques for extraction of BPAs from process models as well as the generation of process models from BPAs. Our BPA methodology's aim is to improve consistency between the different layers of the process model repository and keep maintenance of the different layers at its minimum. Depending on the type

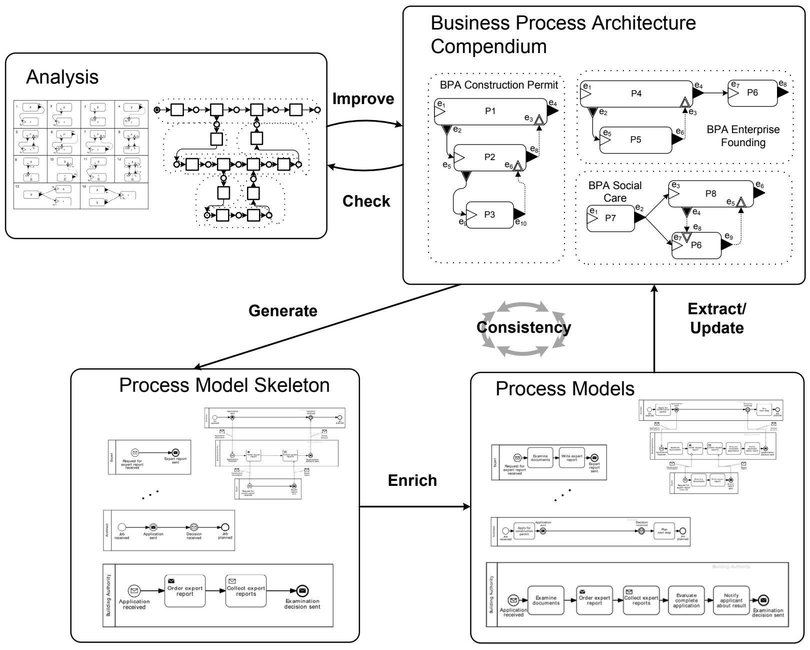


Figure 36: BPA framework

of BPM project our BPA can be used top-down or bottom up. Figure 36 shows our methodology and its different elements. The main element is our BPA approach, represented by the BPA compendium with its BPA subset. It is the start of the top-down use of our BPA approach and the target of the bottom-up use of our BPA approach. The top-down use of our BPA approach consists of four steps, (1) defining the BPA compendium and its BPA subsets, (2) examining them for correctness, (3) decomposing the BPA subsets to process model skeletons, and (4) enriching the process model skeletons into detailed process models. The bottom-up use of our BPA approach consists of the extraction of BPA subsets and hence the BPA compendium. Each element in Figure 36 shows the result of the according step performed, e. g., the decomposition of BPA subset results into several process model skeletons.

7.1.1 *Business Process Architecture Top-Down Approach*

Top-down approaches are often applied in just established companies, or already existing companies that do not have any elicited processes. The aim of the initial phase of the elicitation phase of PA approaches is to generate an ordering structure that can be used for grouping the business processes. The result in many cases are folders with lists of business process names or similar categories that incorporate some categorization logic. The processes are then elicited according to some selection mechanism, or cost function, that ranks and prioritizes them for elicitation.

After the prioritization process, the elicitation of the business processes is performed, one after another until all processes (or only the ones in scope of the project) of the organization have been modeled. The relation between the business process architecture level and the business process model level is a "part of" or "is-in" relation and generally only a mapping from process name to process category.

When applying our Business Process Architecture approach top-down, our BPA approach is used to structure the business process model collection and its inherent processes into groups of interconnected processes. In the elicitation phase the users first define scenarios (BPA subsets). This step results in a BPA compendium that provides an overview of all scenarios (BPA subsets) as depicted in Figure 36. For each scenario the users identify the business processes that are required for the realization of the scenario. As a second step, the interdependencies between processes in a scenario are identified by defining output and input relations or data relations. Following this, the trigger and information flow relations between events of processes are modeled depicting the interaction, input/ouput or data relations between the interdependent business processes. One of the aims of applying our BPA concept *top-down* is to depict and examine interdependencies between business processes early in the design phase. In this way the process errors and problems are detected as early as possible and the process interdependencies are harmonized. Not detected errors deriving from bad process interdependencies cause cost and efforts to correct, especially when many process models are affected.

The formal underpinning of our BPA concept and its analysis techniques allow examining the correctness of the BPA design at a very early stage. Problems found can be improved and subsequently the BPAs can be re-examined for correctness until a correct BPA design has been established. Chapter 6 has presented the different analysis techniques for BPAs. Only if the BPA is correct, the concrete modeling of the operative business processes and their interaction should be performed.

To support the user with the modeling of harmonized business processes under consideration of their interdependencies we present a BPA to process model transformation in Section 7.3. The design of a BPA spans an overarching frame over the underlying business processes. This structure depicted by the BPA is used to reduce modeling efforts by our transformation technique. The transformation generates a process model skeleton for each process in a BPA subset. The process model skeleton exposes the interdependencies defined on BPA level, such that the interaction between processes is harmonized. Based on the process model skeleton the users can now enrich the process model to the desired granularity under the condition that the predefined interaction structure is not violated. To assure this, users are only allowed inserting new nodes or process model components that do not take part in a message flow or depict other process interdependencies. These components can be inserted between the process model skeleton nodes. The result of this course of action are consistent process model, BPA subset, and BPA compendium layers and fully developed process models. The top-down application of our BPA approach guides the users in the process elicitation process and provides support through the BPA to process model skeleton transformation.

7.1.2 *Business Process Architecture Bottom-Up Approach*

The elicitation of business processes in a company is not always planned top-down. Often the need for business process management arises in one department because of pressing issues like budget cuts, demographic change, or bigger market competition. Other departments notice these undertakings and slowly a BPM community develops before BPM is adopted by the company's managers as long term strategy for the optimization of their company's operations. In such rather unstructured BPM approaches, business process models are rather grouped along departmental lines or not having any ordering systems.

In situations where a business process collection already exists, an overarching ordering system for managing and structuring them has to be created ex post. In Section 7.2 we describe a BPA extraction technique that generates BPAs from a business process collection. For this the process models are scanned for elements that take part in message flows or form another type of interdependency. The process is divided into two steps: first, the transformation of each business process model into a BPA process; second, the mapping of message flows, data dependencies, and other concepts that depict an information flow between two or several business processes to BPA trigger and information flow relations. The extraction of a BPA from a process model collection depicts the bottom-up application of our approach.

The extraction algorithm and the bottom-up consistency criteria provide the foundation to generate a topical overview of the interdependencies between processes found on process model level on BPA level. The combination of both top-down and bottom-up approaches and our analysis techniques form our overall BPA framework. Both approaches support the maintenance of the process model collection.

7.1.3 *Maintenance of a Process Model Collection and Business Process Architectures*

The maintenance of a process model collection with hundreds or thousands of business processes is a tiresome and time consuming task. In regular intervals process owners evaluate if the process models in their process repository still reflect the actual operations of the company.

To support the maintenance of process models and the inherent maintenance of the BPA both the top-down and bottom-up approach can be applied. Both approaches complement each other if processes in the process model collection need to be changed as depicted in Figure 36.

When already both process models and BPAs exist, keeping the process models and BPA up to date is a major difficulty. Both the bottom-up and top-down approach that we present in the next sections provide consistency criteria that the transformations and the resulting views, BPAs and the process model skeletons, must comply to. These consistency criteria can be used for the maintenance of the process models and the BPA structure of the process model collection.

Changes, improvements, and updates of process models are decided either on BPA level or on process model level. For instance, in the course of a process re-engineering project or some business process optimization project, a process architect changes the interdependency between processes by inserting new processes to the BPA, adding, removing, and changing trigger or information flow relations. Those changes require changes on the process model level, i. e., new process model skeletons need to be generated, and the changed relations need to be transferred to the already existing processes. The top-down consistency criteria provide pointers to the process models that reveal inconsistent structures, e. g., message flows that do not map to any information flow or trigger relation on BPA level. These can then be removed or changed until the process model layer and the BPA layer are consistent again.

In other situations, the change, improvement, and update is directly performed on process model level, e. g., when a process owner updates his process models due to operational changes. In general there are many process owners that perform such steps so that keeping the BPA topical is rather difficult. The changes made on process model level might need to be propagated to the BPA level in case they cause a change in an interdependency between business processes. To keep the BPA updated, it has to be extracted from the process models again which can be done with the bottom-up extraction described in Section 7.2. In this way old and new BPA can be compared or changes highlighted visually. The newly extracted BPA shows a topical picture of the interdependencies between the business processes within the process model collection.

The middle part of Figure 36 shows how the presented techniques and methodology support the maintenance of a process model collection. Depending from where the change in the process model collection is triggered, the according steps are performed while consistency between both the process model and BPA layer is considered. In the following we present an overview of elements that describe business process interdependencies in process models and the techniques for BPA extraction and process model skeleton generation.

7.2 BPA EXTRACTION ALGORITHM

In this section we present the different elements of process models that are involved in the interaction between process models and show their interdependency. These are required to extract BPAs from process models.

7.2.1 *Process Interdependencies in Process Model Collections*

Process model collections often contain hundreds or thousands of process models that describe the operations of a company [165]. The production of goods or the delivery of services often is the result of a collaboration between several processes that are loosely connected through message flows or data interdependencies.

Interdependencies between process models can derive from several elements in a process model. Some interdependencies are explicitly modeled such as message flows, other may be hidden, e. g., data dependencies. For this we especially look at the two most commonly used process modeling notations Event-Driven Process Chains (EPC) and Business Process Model and Notation 2.0 (BPMN 2.0) and present common structures and their mappings.

In the following we introduce different process modeling language independent aspects that depict trigger and information flow interdependencies and then provide notation specific elements and constructs that form those interdependencies in Section 7.2.2 and Section 7.2.3.

The most obvious interdependency between processes is the exchange of information between two processes, e. g., a message flow that has a source, the sending element, and a sink, the receiving element. Here, one process sends information that another process requires for further execution. The sending and receiving of information can be depicted in different ways in process modeling languages. First of all, different node types can be involved, e. g., activities, functions or events. The message flow may be visualized or only depicted by interfaces or labels that match.

Trigger and information flow relation can be reduced to a simple interdependency between a sending element and a receiving element whereas both elements are part of different processes. On the very basis we need to identify process model elements that depict the provisioning of information, and others that show the consumption of information.

To differentiate between information flow and trigger, the position of the receiving element in a process is important. In trigger relations this is the element that starts a process. In this regard all elements that depict the reception of information in the beginning of a process are of interest. In an information flow relation the receiving element has to be located somewhere between process start and process end, i. e., the elements that show the reception of information during process execution are of importance.

In general we have to identify all kinds of elements that can consume or provide information in a process model. These can be interfaces, inputs like forms or data objects, active elements like activities or functions, or descriptive elements like events or links to other processes. The start of a process is often depicted by a particular event that instantiates the process upon its occurrence. Similarly, the termination of a process is often described by an event, e. g., something has been produced or a particular result has been achieved. Of course the start and termination of a process is described in each process modeling notation differently. This may even involve a particular node type for the start or the end of a process.

Besides message flows, data objects play are particular role. An interdependency between two processes may derive from the use of the same data objects where one process induces a particular data object state that is the input to another process. We will enlighten the specific characteristics of data interdependencies in Chapter 8.

Roles or organization elements are often involved in different processes, however, they do not qualify for extracting process interdependencies as they do not provide any detail when the information is sent. Despite that, roles could rather be used for providing another view on process groups that involve the same role or to extract the interaction between roles as done by choreography diagrams. Hence, we focus only on sending and receiving elements, and data interdependencies in our examination.

Interdependencies between processes in a process modeling collection can be grouped for each process modeling notation into two different categories, explicit interdependencies and implicit (hidden) interdependencies. Explicit interdependencies are visualized in a process model explicitly, e. g., by an association or message flow arc. Implicit interdependencies are not visualized and are only found when analyzing the process models in a process model collection for shared elements or other commonalities, e. g., the use of the same data object or matching element labels. Explicit and implicit interdependencies hence differ for each modeling notation depending on the capabilities and focus of the process modeling notation.

7.2.2 *BPMN 2.0 Model Collections*

BPMN 2.0 has become a de facto standard for BPM. It eases the modeling of complex business operations with a variety of elements [32]. BPMN collaboration diagrams explicitly specify sending and receiving elements, i. e., tasks, events, and pools. Here pools refer to a process that encompasses tasks and events and depict the outer shell of a process. For the BPA extraction we focus on the inner ingredients of a process; tasks and events. Collapsed pools cannot be used for the extraction of PAs as the order of occurrence of those elements is hidden and cannot be extracted.

In BPMN 2.0 the interaction between processes is often depicted by events, e. g., "document received", or by activities, e. g., "send documents" that are connected by a message flow, a dotted arc. More hidden, but also commonly found, the data access of processes depicts an interaction between processes in regard to writing and later reading of the same data object. Finding those hidden data interdependencies, however, is rather complex. Chapter 8 elaborates on extracting data BPAs based on found data interdependencies.

Figure 37 shows BPMN collaboration diagram elements that are involved in forming process interdependencies. It depicts various BPMN start, intermediate, and end events, data objects, and different task types, as well as the different flow arcs that are used in a BPMN diagram to differentiate between sequence, message, and data flow.

Our BPA concept can be used with process models that use message or signal events, as well as send and receive tasks for depicting interaction between processes without the need of any specific mapping. Other event types could also be mapped to our BPA concept but need further analysis for the development of an extraction handler. This is content of future work. Especially the handling of error or escalation events that lead to the interruption or abortion of a process need thorough examination. For the moment our extraction

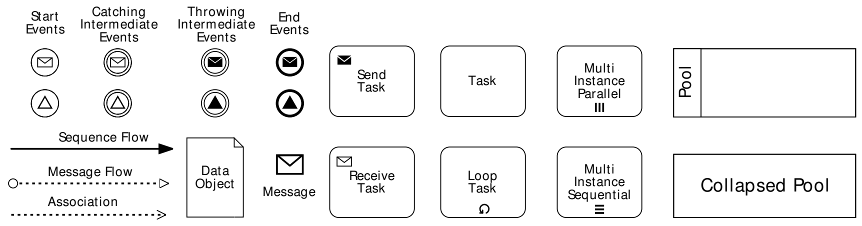


Figure 37: Selected BPMN elements

algorithm excludes escalation, error, compensation, link, conditional, (parallel) multiple, and timer events. It handles normal, message, and signal events, which represent a set of events that are commonly used.

Start and catching intermediate events can be grouped as receiving events, that receive a trigger (start event) or some information (intermediate catching event) from an external source, i. e., another process. Throwing intermediate events and end events can be regarded as sending events that send information to other processes. In Figure 37 only message and signal events were depicted as a proxy of the various sending and receiving event types of BPMN 2.0. In each case the events are either source (sending events) of a message flow or the sink/target (receiving events) of a message flow. BPMN intermediate events usually are only in 1:1 relations with other events, i. e., a sending event only sends a message to one receiving event. Signal events broadcast their signal to all catching signal events that are interested in that signal. Message end events can have several outgoing message flows, i. e., several targets. Message start events can have several incoming flows, each depicting upon reception the instantiation of the process [112].

BPMN 2.0 defines specific sending and receiving task types that explicitly show that the according task sends, respectively receives, an information from another process. However, in many cases normal tasks with a message flow are used to depict the sending and receiving of information, e. g., labeled with "send documents". For the extraction it is important to either detect an incoming or outgoing message flow or the according task type, or both. As there are several modeling styles, this is rather a matter of configuration. Send and receive task, in contrast to message events, can be in a 1:n relation with communication partners. A receiving task can receive information from different sending nodes according to the BPMN2.0 specification [112]. However, if a receive task has several incoming message flows only one flow is applied to one instance of the task. Several message flows could only be handled if also several instances of the tasks are instantiated by the incoming sequence flow of the receiving task. In contrast to the receive task, a send task that has several outgoing message flows sends a message to all its receiving nodes

BPMN differentiates between sequence flow (the internal control flow) and the message flow that depicts the interaction with other processes. The association flow is used to define the relation between activities and data objects. Chapter 8 discusses the extraction of data dependencies from BPMN2.0 process models.

We present a few exemplary interaction structures that are the basis for extracting BPAs from a BPMN process model collection. Figure 38 shows common process structures in BPMN models that reflect interdependencies between processes. Of course, process interdependen-

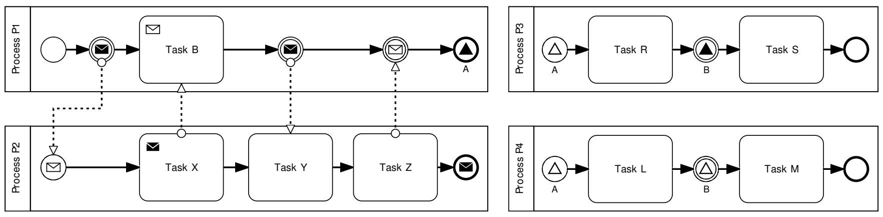


Figure 38: Exemplary patterns found in BPMN process models

cies are not restricted to those structures and can result from further process interaction structures or data interdependencies.

In Figure 38 we see four process models p_1, p_2, p_3, and p_4. Processes p_1 and p_2 depict different ways of message exchanges. First, process p_1 sends a message to process p_2 via an intermediate throwing event. Process p_2, receiving it via its start event, is instantiated by this message. Hence, p_2 can only start after p_1 has started and has sent the first message to p_2. The information exchange between both processes continues by a combination of sending and receiving tasks, and sending and receiving events. After p_2 has started, sending task x sends a message to p_1 which is received by the receiving task b. The other information exchanges show a combination of an untyped task with a message flow either receiving or sending some information from or to an event of process p_2.

The BPMN specification does not demand for send and receive tasks, or message events, that their incoming, respectively outgoing message flow is displayed. Hence, a message event or send or receive task without message flow must specify their communication partner in a different way, e. g., in the meta data of the process elements. We assume that this information is available for extracting BPAs.

The situation is similar for signal events. E. g., looking at the signal end event of process p_1 one can hardly notice from the process model that it is connected to the start signal events of processes p_3 and p_4. This is only possible by matching their labels. In Figure 38 process p_1 broadcasts a signal and instantiates both processes p_3 and p_4. We consider such relations as a special kind of message flow, although they are hidden in the visual representation. For the BPA extraction we assume them to be given either in their labels or in the definition of the signal events.

All tasks and events that partake in a message flow can be regarded either as sending or receiving node for the BPA extraction. Hence, each of the BPMN process models has to be examined for those nodes associated with message flows, displayed visually or defined in the node definition.

7.2.3 *EPC Model Collections*

The Event-Driven Process Chains process modeling notation has less symbols and in this regard appears to be more simple than BPMN. It is rather underspecified and lacks well defined syntax and semantics [148]. The lack of clear semantics poses difficulties for the extraction of BPAs from EPC models. The focus of EPC is the display of the dynamic sequence of activities and events [129].

Figure 39 presents the main symbols of the EPC notation. The EPC elements are events, functions, roles, and a group of symbols representing data objects. In general, the focus of EPCs is to model rather single processes than process collaborations. Because of that, EPCs have only few symbols for expressing process interaction. In EPCs there is no specific message flow symbol, hence in many cases the interaction between processes can only be detected by the data objects, events, labels of activities, or the process interface symbol. The process interface element can be considered as a special event that instantiates other processes and in this way also expresses an interdependency between processes. It is the only element providing a mean to directly express the interaction between two processes. Interdependencies between EPC process models are rather hidden in labels or data objects. The control flow is depicted by arcs connecting functions to events and vice versa. Each function and each event is allowed to have only one incoming and/or one outgoing edge. Neither two functions nor two events may follow each other on a control flow. Function and event must be alternating on a control flow path [129].

For detecting interdependencies events and functions play a major role. Events describe the occurrence of a significant incidence or the change of a condition. They characterize the ending and result of an activity that may again be the characterization of the start of the next activity [129]. A complete EPC process describes the creation of an output which can then be again the input of another process. These relations between output and input of processes are described by their events. Each EPC process model generally starts with an event node describing the condition or the event that instantiates the process [25]. The end of a process is described by an event as well. However, events can be also compound to more complex structures. Decker at al.[25] elaborate on the complex instantiation structures of EPC. Their complexity was visualized during the application of our pattern and antipattern analysis of the SAP reference model in Section 6.1.

EPC do not provide an inherent mean to identify intermediate sending and receiving events. This in return makes the BPA extraction without any assumptions difficult. Events with only one outgoing flow and no incoming flow that are meant to occur before any activity are considered start events. Events with one incoming and one outgoing flow that occur after an activity has already occurred and before all activities have occurred are considered intermediate events. However, as EPC are bipartite and a function can only have one incoming event flow, additional intermediary events cannot be attached that would express the sending or receiving of information. Using gateways for creating compound events does not fully solve this problem as the additional events attached in the middle of the process would rather be considered start events as well. Especially, if the compound event is attached by an XOR gateway. We disregard this possibility as we assume EPC process models to have only one start and one end event. Compound events that are connected to a function by an AND gateway from which one event is an intermediary event will be considered intermediate events for the extraction of BPAs. This is a hardly satisfying solution as it bears further complexity for analyzing those compound events. OR gateways are disregarded as they cover both possibilities. An end event is an event that has only one incoming flow and that can only occur after all activities of the process have occurred.

We assume that intermediary events (not compound) that are connected to two functions by an incoming and one outgoing flow send or receive information if they match labels with an event of another process model. In this case, we assume that there is an interdependency. The difficulty is to determine the direction of the information exchange as both could be initiator of the information exchange. We assume that the direction of exchange can be identified and the sending and receiving event determined, for example by analyzing the labels for sending or receiving vocabulary with a specified glossary or by analyzing attached data object states. EPC function elements are the equivalent to BPMN activities. They are active elements and show that an activity is performed in the process. Like events, they can be used to depict external communication by their label, e. g., when labeled "receive documents" or "send order". But also here sending and receiving of information can only be determined by the analysis of labels as EPC do not provide explicit message flows. Similar to BPMN, data objects in EPC are another source for detecting interdependencies between processes. Figure 39 shows different symbols representing data in the EPC modeling notation. These are the elements "Form", "Data Object", "Email", "Letter", and "Text Note" on the right hand side of Figure 39. Nonetheless, the association of data objects to function is undirected in EPC, so that determining if a data object is input or output of a function is not possible from only looking at the associa-

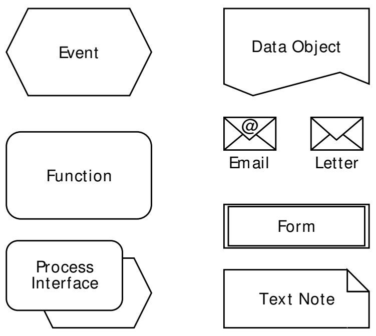


Figure 39: EPC elements

tion. The analysis of the data and function labels could give insight for determining the function of the data object. Such information may also be provided in the according meta data of the process element. For the BPA extraction from EPC models we do not consider data objects as the handling of data objects would require advanced natural language processing algorithms.

Hence, events and process interfaces depict the main source for detecting interdependencies between processes in an EPC process model collection. Functions only provide information on interdependencies through their label but do not provide any structural information for interdependency detection. We assume that functions that send to or receive information from another process can be identified and this information is given for the extraction of BPAs. Matching start events (input), end events (output), and particular structural constructs of intermediary events depict an information exchange between processes. We regard them as message flow and assume them to be available as well when extracting BPAs from EPCs.

7.2.4 *Basic BPA Extraction Algorithm*

The extraction of BPAs from a process model collection is performed in two steps. First, the process models are transformed to BPA processes. Second, after detecting the various elements that partake in interdependencies and interactions between the processes on process model level, the detected interdependencies are mapped to trigger and information flow relations and inserted into the BPA. The result is a BPA showing the content of the process model collection in an abstract way and highlighting the complex interdependencies, e. g., for process optimization and restructuring efforts. The extraction algorithm shall retain the interdependencies found between process models in regard to their initialization and show the strength of interdependence (the number of interactions). We will illustrate the extraction algorithm on our simplified construction permit scenario from the public administration that is shown as BPMN 2.0 processes in Figure 40.

In the following we introduce the definition of a process model collection, that we denote as PMC, and the message flow relations that depict the interdependencies between processes in a process model collection.

Definition 43 (Process Model Collection) *Let* $\mathcal{PM} = (PM_1, ..., PM_n)$ *be a set of process models. The tuple* $PMC = (\mathcal{PM}, K, \mathbb{F})$ *consists of a set of process models describing a process model collection in which*

- PM_i *with* $1 \leq i \leq n$, *depicts each process model in* PMC, *the process model collection.*

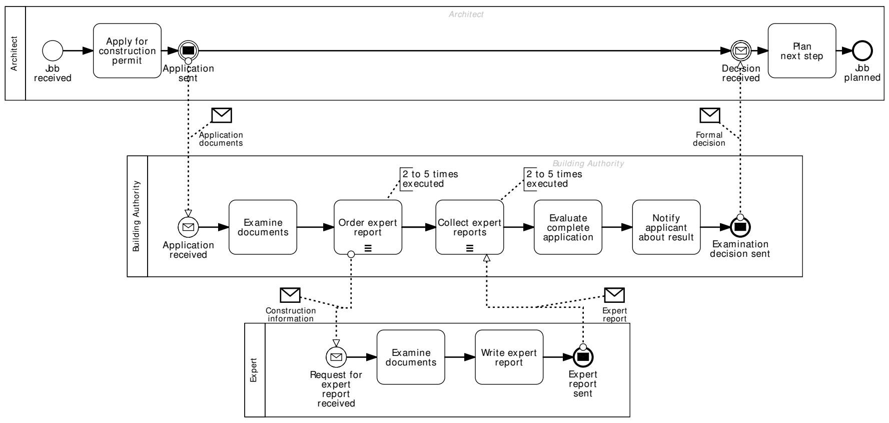


Figure 40: Simple construction permit application

- $K_i \subseteq A_i \cup E_i$ *is a subset of nodes consisting of only activities and events that take part in the interaction between two or more processes.*
- $\mathbb{F} \subseteq CF \cup MF$ *is the overall set of flows in the process model collection where* CF *and* MF *are disjoint.*
- $MF_{i,j} \subseteq K_i \times K_j, i \neq j$ *describes the message flow relation between two process models where* $(k_1, k_2) \in MF$ *with* $k_1 \in PM_1$ *and* $k_2 \in PM_2$*, i. e., the nodes belong to two different process models.* ⋄

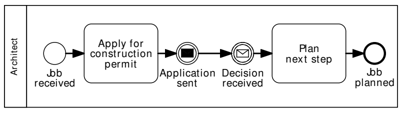
Based on the definition of a process model collection, we first describe a simple extraction algorithm that assumes sequential process models in the following section. After that, we show a more complex extraction algorithm that allows for the extraction of BPAs from more complex block structured process models in Section 7.2.5.

The BPA extraction algorithm is based on the formal normalized process model definition presented in Definition 4. It considers activities and events of a process model for extracting BPAs. The basic algorithm applies to process models that are sequences.

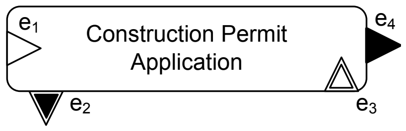
The abstraction concept of the basic algorithm can be categorized by abstraction by elimination according to [136] as we eliminate insignificant elements and keep only the ones of interest. The extraction algorithm starts with the extraction of significant elements of a process model.

EXTRACTION OF BPA PROCESSES FROM PROCESS MODELS. For the extraction we need to detect the elements in a process model that interact with other processes that were introduced in Section 7.2.1, Section 7.2.2, and in Section 7.2.3. These can be subsumed in nodes being activities and events that take part in a message flow. A message flow shows the exchange of information between two processes and subsumes various ways of interactions between different elements discussed in previous sections.

Structurally, each process model has one start and one end node that can be an activity or an event. Those nodes do not have a preset or postset respectively in regard to their control flow. The first node of a process model being an event or activity maps to the start event of the BPA process. Similarly, the end node of a process model maps to the end event of the BPA process. Figure 41 shows the exemplary extraction of a BPA process from a process model. All nodes that are not start or end nodes and do not take part in a message flow are eliminated from the process model. Of all the intermediary nodes of the process model only those nodes that take part in a message flow will be mapped to an according BPA process event. The intermediary nodes can be detected by analyzing their pre- and postsets and if they are the source or the target of a message flow. E. g., in Figure 41a the intermediate throwing event "Application sent" of the process is mapped to the intermediate throwing event e_2 in the BPA process. The intermediate nodes, i. e., the task "Apply for construction permit" and the task "Plan next step", are not considered and hence not extracted to the BPA process. In general all intermediate sending nodes (events, activities) will be mapped to an intermediate throwing event of a BPA process. In a similar way, receiving elements are mapped to intermediate catching events of the BPA process, e. g., the catching intermediate event "Decision received" is mapped to the intermediate catching event e_3 of the BPA process.



(a) BPMN Process



(b) Extracted BPA Process

Figure 41: Extraction of BPA process

All other nodes that are neither a start nor an end node, nor an intermediary node taking part in a message flow, are ignored and not represented in the BPA process as only elements that depict interaction are of interest. We refer to nodes of interest as *significant* nodes [136]. As the basic extraction algorithm is only applied to process models that are sequences and no loops are allowed, each node in a process is only executed once. Hence, we assign a trivial multiplicity to the event in the BPA process, unless the original source node has multi-instance or looping attributes, e. g., BPMN multi-instance task or loop task. Then the multiplicity set of the according event in the BPA process is set to the possible execution times specified in the attributes of the original process model node. The trivial multiplicity is not visually depicted in a BPA process diagram.

Each extracted BPA process is understood as a sequence of events that are interconnected by information flow and trigger, depicting their process interaction. The interaction found on process model level needs to be mapped to two concepts on BPA level, trigger and information flow.

EXTRACTING INTERDEPENDENCIES BETWEEN PROCESS MODELS. After having described the extraction of processes in the process model collection we need to add triggers and information flows to connect the extracted BPA processes on BPA level. The interdependencies between processes can be attributed to the relations between specified elements in the process models introduced in previous sections. These interdependencies are mapped to message flows on process model level. In BPMN those interdependencies mainly derive from message flows that connect combinations of activity and event nodes of the interacting process models. We consider matching signal events in BPMN, and matching events and process interfaces in EPC as message flows in our extraction formalism that we present in the next paragraph. All message flows between process models that have a start node as receiving partner are mapped to a BPA trigger. All message flows between process models that have an intermediate node as receiving partner are mapped to a BPA information flow.

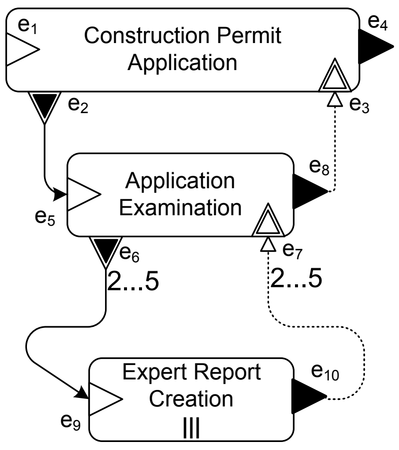


Figure 42: Extracted BPA construction permit application

For each message flow pair found in the process model collection we introduce the according trigger or information flow in the BPA by connecting the according sending and receiving events of the BPA processes. Figure 42 shows the extracted BPA, our running example, from our general simplified scenario in Figure 40. Each process model was transformed to a BPA process that only consists of the elements that interact with other processes. The start node and end node are always extracted to a BPA process as they show the instantiation and termination of a process. All other nodes, e. g., tasks that are only performed internally are eliminated and not extracted to the BPA level, e. g., the "Examine documents" task of the building authority process was not transferred to BPA level as it concerns no other process. All interdependencies in form of message flows were transformed to either triggers or information flows on BPA level. E. g., the sending activity "Order expert report" that targets the message start event of the "Expert report" process was transformed into a trigger in the BPA. The multiplicity annotation of process two was reflected in the multiplicity set of the according events.

The following definition describes the extraction of BPA processes from process models in a formal way.

Definition 44 (Process Model to BPA Process Transformation) *Let* $nPM = (N, CF, \mathit{type}, s, f)$ *be a sequential process model of a process collection* PMC. *Let* $nPM' = (N, CF, \mathit{type}, s, f)$ *be a second process model that* nPM *interacts with. Let* $\nu =< e_1, e_2, ..., e_m >$ *be a BPA process that maps to the process model* nPM. *The transformation from a process model into a BPA process is defined as follows:*

- $\nu = \{e_1 \in E^S | n = s \wedge n \in N \setminus G \wedge \bullet n = \emptyset\} \cup$
  $\{e_i \in E^T | n \in N \setminus G \wedge n\bullet, \bullet n \neq \emptyset \wedge (n, n') \in MF, n' \in nPM'\} \cup$
  $\{e_i \in E^C | n \in N \setminus G \wedge n\bullet, \bullet n \neq \emptyset \wedge (n', n) \in MF, n' \in nPM'\}$
  $\{e_m \in E^E | n = f \wedge n \in N \setminus G \wedge n\bullet = \emptyset\}$
- $mult(n) = \mu(e)$ *denotes that the multiplicity of a BPA event is equal to the execution times of the process model's node.* ◇

Definition 45 (Process Model Interdependency to BPA Mapping) *Let* $\mathcal{PM} = (nPM_1, ..., nPM_n)$ *be a set of process models and let* $PMC = (\mathcal{PM}, MF)$ *be a process model collection. Let BPA* $= (E, V, L, I, \chi, \mu, \pm)$ *be the according process architecture. The function* $ret : N \to E$ *retrieves for a node in a process model the matching event in the BPA. The function* $mapflow : MF \to L \cup I$ *maps the message interaction of process model level to trigger and information flow on BPA level, such that* $mapflow((n, n')) = (ret(n), ret(n')) \in I \cup L$. ◇

In the following we describe the consistency criteria between BPA and process model layer of a process model collection in a formal way. In summary, the formalism requires that each BPA element of a process collection's BPA must have a partner element on the lower more detailed process model layer.

Definition 46 (Consistency Criteria) *Let* $\mathcal{PM} = (nPM_1, ..., nPM_n)$ *be a set of process models. Let* $PMC = (\mathcal{PM}, MF)$ *be a process model collection. Let*$\mathcal{A} = \bigcup_{nPM_i \in \mathcal{PM}} A_i$, $\mathcal{E} = \bigcup_{nPM_i \in \mathcal{PM}} E_i$. *The according BPA* $= (E, V, L, I, \chi, \mu, \pm)$ *needs to comply to the following consistency criteria.*

- $E^S = \{n \in (A_i \cup E_i) : \bullet n = \emptyset \wedge n\bullet \neq \emptyset\}$
- $E^T = \{n \in (A_i \cup E_i) : \bullet n, n\bullet \neq \emptyset \wedge \exists b \in A_j \cup E_j, i \neq j : (n, b) \in MF\}$
- $E^C = \{n \in (A_i \cup E_i) : \bullet n, n\bullet \neq \emptyset \wedge \exists b \in A_j \cup E_j, i \neq j : (b, n) \in MF\}$
- $E^E = \{n \in (A_i \cup E_i) : \bullet n \neq \emptyset \wedge n\bullet = \emptyset\}$
- $I = \{(n_1, n_2) \in MF : \nexists n_3 \in N : (n_3, n_2) \in CF\}$
- $L = \{(n_1, n_2) \in MF : \exists n_3, n_4 \in N : (n_3, n_2) \in CF \wedge (n_2, n_4) \in CF\}$
  ◇

The setting of the conflict relation χ between flows of several processes and the setting of the correspondence relation requires additional information captured in more complex process model structures like XOR-bonds or data objects. The complex extraction and data BPA extraction algorithms presented in Section 7.2.5 and in Chapter 8 make use of the χ relation to reflect those constructs on BPA level.

For the extraction the BPMN 2.0 collaboration diagrams and EPC process models need to be mapped to the formal and more abstract process model definition in Definition 4. Tasks in BPMN and functions in EPC are mapped to activity nodes of the process model. Events of both BPMN and EPC are mapped to event nodes of the process model, gateways to gateway nodes which are typed by the according gateway type. The control flow of the BPMN and EPC models is transferred as is to the formal process model definition. Depending on the definition of message flows the according elements must be identified and message flows associated to the according elements in the normalized process model.

Before transforming a BPA from a process model collection the target domain needs to be defined, i. e., whether the BPA represents interactions in a synchronous or asynchronous environment. According to these characteristics only the basis algorithm or the complex extraction algorithm is applied. Synchronous environments require order preserving extraction which is provided by the basic extraction algorithm. The basic algorithm results into BPA processes that are trace equivalent with their source process models in regard to their external behavior. I. e., we observe the same order of incoming and outgoing message flows of a process model on process model and BPA level. The complex extraction algorithm is not internal flow order preserving. Hence it can only be applied for asynchronous interaction environments. The characteristics of the target environment cannot be determined by examining the process models and need input from a business process expert. The basic extraction algorithm allows applying our BPA analysis techniques without restrictions.

7.2.5 *Complex Extractions*

Section 7.2.4 introduced a basic algorithm that generates BPAs from process models that represent a sequence. Based on that, we introduce an extension of the extraction algorithm for dealing with more complex process models. We assume our process models to be block structured, free of rigids, and structurally sound, i. e., they have exactly one start and exactly one end node, and all nodes are on a path from the start node to the end node. Our extraction algorithm resorts to research on process model abstraction based on RPST_{PM} by Smirnov et al. [135, 136, 141] and Vanhatalo [159, 160]. Smirnov et al. [135, 136, 141] focus on single processes and consider activities of process models in their extraction algorithms. We take a broader view to extract interdependencies between several processes and we consider activities and events for the extraction of BPAs from process models as they are our main source for detecting interdependencies. The basic techniques and formalism for decomposing process models into an RPST_{PM} (see Definition 10) were introduced in Section 2.1.1. The decomposition of a process model based on RPST_{PM} results into a hierarchical tree of RPST_{PM} components. The RPST_{PM} components on the first level of the RPST_{PM} are direct children of the RPST_{PM} root node. The RPST_{PM} components on the first level of the RPST_{PM} form a sequence on which we can apply our basic extraction algorithm. The idea of our algorithm is to aggregate all other RPST_{PM} components (direct and indirect children of the first level nodes) into sequences of events on the first level of the RPST_{PM}. In contrast to our basic BPA extraction algorithm, the complex algorithm uses concepts from abstraction by elimination as well as abstraction by aggregation. In this regard, the complex extraction algorithm, shall also retain the causal interdependencies found between process models in regard to their initialization relation and show the strength of interdependence (the number of interactions). Hence, we allow that the exact order of sending and receiving nodes that belong to one component on process model level may differ on BPA level, i. e., the BPA representation is not trace equivalent to the external behavior of the process model.

In the first step, we eliminate all nodes that do not take part in an interdependency, i. e., all nodes that are not part of a message flow on process model level. However, start and end nodes that are not part of an interdependency are preserved. This leaves us only with *significant* nodes that, if required, are aggregated. We call the nodes that are part of an interdependency *significant* [136]. Aggregation is a many-to-one mapping where several elements are aggregated to one element that represents them all. Using this concept we aggregate several significant process model nodes and map them to one event in a BPA process. Our extraction algorithm consists of four steps:

1. The creation of an RPST_{PM}
2. The elimination of insignificant nodes
3. The aggregation of components up to the first level components of the RPST_{PM}
4. The transformation of aggregated components (direct children of the root node) and definition of flows and multiplicities

CREATING AN RPST_{PM}. The decomposition of a block structured process model into an RPST_{PM} provides us on the root level (level 0) with one block, a polygon (PO), that contains all process model fragments. On the first level of the RPST_{PM} we receive a sequence of canonical components that either are trivial components (TR), polygons (PO), bonds (BO), or rigids (RI). Polygons are sequences of nodes or sequences of components.

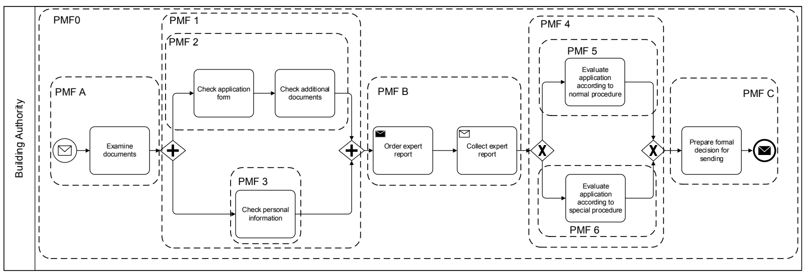

Building Authority
PMF0
PMF 1
PMF 2
PMF A
Examine documents
Check application form
Check additional documents
PMF 3
Check personal information
PMF B
Order expert report
Collect expert report
PMF 4
PMF 5
Evaluate application according to normal procedure
Evaluate application according to special procedure
PMF 6
PMF C
Prepare formal decision for sending

Figure 43: Business process model decomposed into $\mathrm{RPST}_{\mathrm{PM}}$ fragments

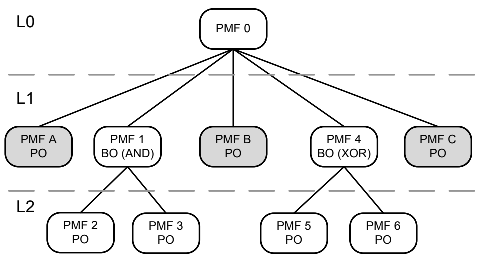


Figure 44: $\mathrm{RPST}_{\mathrm{PM}}$ of original business process model

Bonds are formed by a set of components that share the same boundary nodes, i. e., entry and exit nodes. Rigids (RI) are components that cannot be classified into any of the other categories. We assume that rigids do not contain nodes that depict interdependencies.

Figure 43 presents an exemplary process model that is decomposed into process fragments defined by its $\mathrm{RPST}_{\mathrm{PM}}$ decomposition. The process model fragments containing significant nodes are named with letters, e. g., $\mathrm{PMF}_{\mathrm{A}}$ or $\mathrm{PMF}_{\mathrm{B}}$. The according $\mathrm{RPST}_{\mathrm{PM}}$ is depicted in Figure 44. We do not list trivial components in the $\mathrm{RPST}_{\mathrm{PM}}$. The components in the $\mathrm{RPST}_{\mathrm{PM}}$ that contain significant nodes are highlighted in grey. The nodes of the $\mathrm{RPST}_{\mathrm{PM}}$ are labeled according to the process model fragments shown in Figure 43 and their type. On the first level of the $\mathrm{RPST}_{\mathrm{PM}}$ there are three polygons with significant nodes and two bonds (one AND-bond, one XOR-bond) that contain only insignificant nodes. The $\mathrm{RPST}_{\mathrm{PM}}$ is the basis for the elimination of insignificant nodes and the aggregation of significant nodes.

ELIMINATION AND AGGREGATION. We eliminate all insignificant nodes of the process model, i. e., nodes (events and activities) that are not source or target of a message flow. The elimination of nodes is performed by removing an insignificant node reconnecting predecessor and successor nodes of the removed node. If all nodes of a process model fragment are insignificant, we remove the complete component in the $\mathrm{RPST}_{\mathrm{PM}}$. For instance, we remove the entire AND-bond PMF_1 in Figure 44 as it does not contain any significant node for extraction. Figure 46 shows the elimination of insignificant nodes and components, e. g., the activity of $\mathrm{PMF}_{\mathrm{A}}$ and the whole AND-bond component depicted by PMF_1 are removed. Figure 45 shows the according $RPST_{PM}$ that highlights the removed components by dashed lines. The result of this step, a reduced process model and the according reduced $RPST_{PM}$, are depicted in Figure 47a and in Figure 47b. Conceptually, we receive an $RPST_{PM}$ with only significant nodes for the BPA extraction. The process model has become a sequence of significant nodes which can be extracted with the basic BPA extraction algorithm.

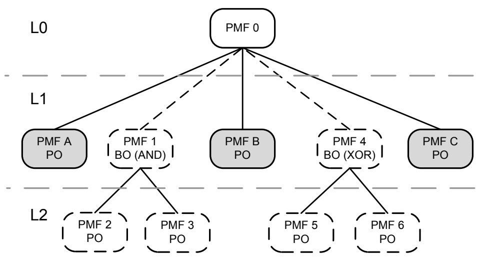


Figure 45: $RPST_{PM}$ of business process model with eliminated nodes

It is obvious that process models can be of any complexity as long as the significant nodes are situated only in polygons that are direct children of the $RPST_{PM}$ root node. In such cases, some components of the process model that contain only insignificant nodes could also be rigids. In general, however, we assume that the process models are free of rigids.

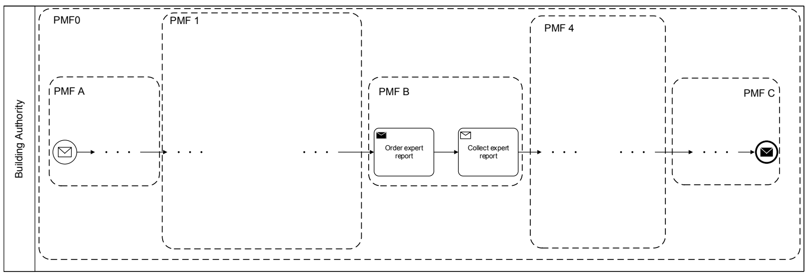
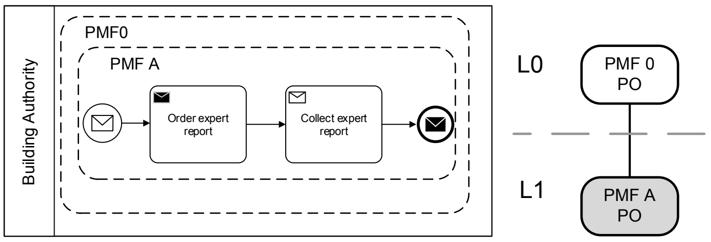


Figure 46: Business process model showing eliminated nodes and $RPST_{PM}$ fragments

In many cases we find XOR-bond and AND-bond components on the first level of the $RPST_{PM}$ that contain significant nodes. In these cases, the elimination of insignificant nodes and components is not sufficient for the aggregation and extraction of BPAs. The significant nodes in those bonds need to be aggregated to a sequential representation up to the first level of the $RPST_{PM}$.

To determine the sequential representation of a component we make use of the containment relation of $RPST_{PM}$ components, e. g., the branches of an XOR-split and XOR-join in a process model are depicted by the direct children of an XOR-bond component. Before we can determine a sequential representation for such a bond component, we need to determine a sequential representation for each of its children. This may be done recursively depending on the depth of the $RPST_{PM}$. Omitting trivial components, the leaves of a $RPST_{PM}$ are polygons, e. g., as depicted in Figure 44. The aggregation of polygons consisting of only trivial components is the basis for determining a sequential representation of a bond component.



(a) BPMN Process (b) RPST_{PM}

Figure 47: Abstracted process model and its RPST_{PM}

AGGREGATION OF POLYGONS. The aggregation of polygons with only trivial components is performed by parsing each node in the component and determining their type (sending or receiving). Neighboring nodes of the same type are aggregated but their references to their partner nodes of the message flow are kept. When aggregating two nodes their message flow preset respectively postsets are added. Depending on their parent component, additional information like their number of occurrences or if they are in conflict are stored in the aggregated node as well. This information is used for the later transformation to BPA events and processes. In the best case, if all nodes in a polygon are of the same type, the result of the aggregation step is one node that has references to many other nodes of other processes. Figure 48 shows different exemplary transformations of process model fragments to BPA constructs. The two transformations of example 1a and 1b in Figure 48 depict an aggregation of node sequences that have the same type. The start and end nodes of a process model are not considered in an aggregation step. We determine the start and end node by employing behavioral profiles. If according to the behavioral profile a node is minimal, it is the start node in the process model; if it is maximal, it is the end node of the process model [163, 162]. In the RPST_{PM} the start node is the leftmost element and the end node the rightmost element. For the aggregation of bond-components several aspects are important:

- the type of an RPST_{PM} bond component
- the number of interaction partners of one RPST_{PM} bond component
- the number of nodes on each branch of an RPST_{PM} bond component
- the existence of one or several types of nodes (sending, receiving, or sending and receiving) in an RPST_{PM} bond component

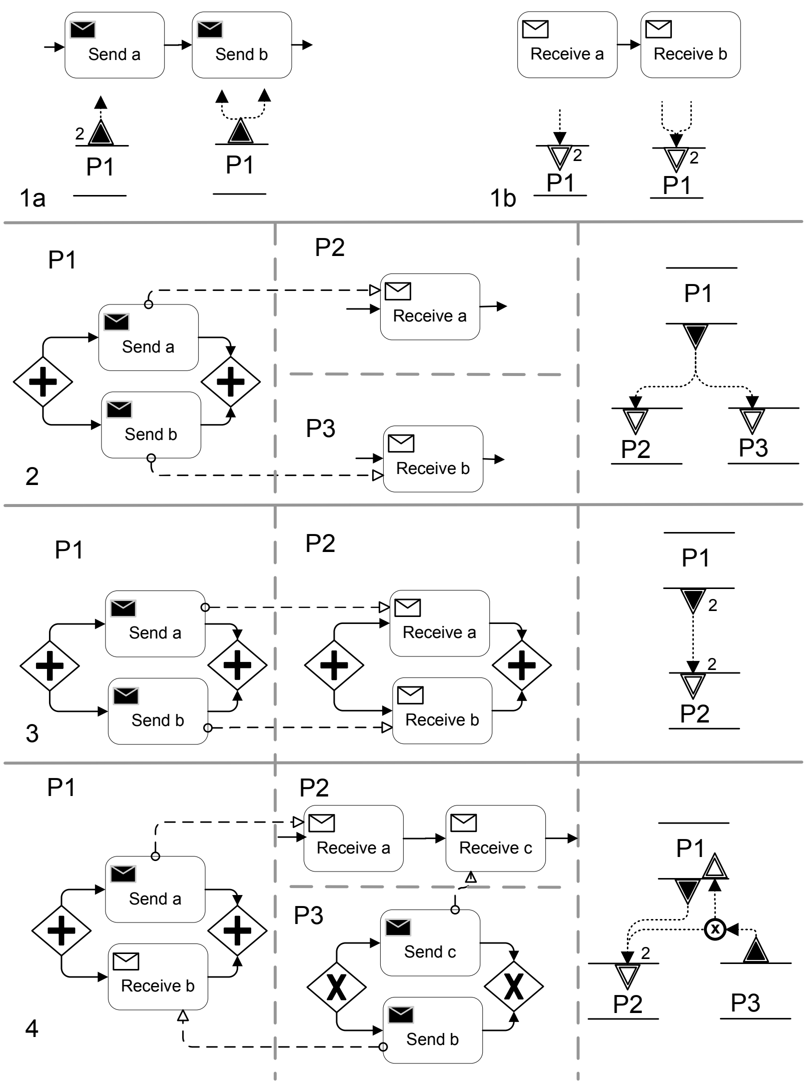


Figure 48: Exemplary process model fragments to BPA transformation

In the following we describe how we deal with each bond component type as different strategies are employed for each bond type.

AGGREGATION OF AND-BONDS. To determine a representation of an AND-bond component its children are compared for their characteristics. The direct children of a bond component that are polygons each depict a branch of the AND block of the process model. Before we compare the branches, we aggregate each branch (polygons) according to the aggregation technique for polygons. The comparison can result in different situations. If all polygons contain the same type of node, all nodes are aggregated to one node of that type. Figure 48 shows the aggregation and the transformation into BPA events of such cases in example 2 and 3. Example 2 depicts excerpts of three processes. The excerpt of process p1 shows an AND-bond that has one sending task on each branch that each sends a message to a different process. The excerpts of processes p_2 and p_3 depict a simple sequence of a receiving task. The AND-bond of process p_1 is transformed into one intermediate throwing event of BPA process p_1, and the receiving events of the other processes each into one intermediate throwing event of their according processes. The sending of two messages on process model level is depicted by the information flow relation between the throwing event of process p_1 with the intermediate catching events of processes p_2 and p_3. Example 3 shows a similar situation between only two processes except that the excerpt of process p_2 shows an AND-bond with a receiving event on each branch. In this case, both aggregations result into one event with multiplicity $\mu(e) = \{2\}$ in each BPA process that are related via an information flow. AND-bonds consisting of nodes of only one type are treated like neighboring nodes on a sequence and can be aggregated. Both examples represent the sending of two messages in different forms.

Example 4 shows an excerpt of a process p_1 that consists of an AND-bond with two branches that have different type of nodes. One branch has a sending node, the other branch has a receiving node. In this case, we cannot aggregate the nodes to one node. The mapping chosen for the AND-bond is a sequence of nodes where the sending node occurs before the receiving node. This is due to the fact that all branches of an AND-bond are activated. A branch that contains a polygon of only sending events will always be executed as the execution is in the control of the process, so that they are independent from other processes. The completion of the whole AND-bond depends only on the receiving events. If a branch contains only receiving events and the other branches only sending events, the whole AND-bond completion depends on the branch with the receiving event as they come from an external source. If the message is not received, the AND-bond exit will not be performed, also if all sending nodes on the other branches have already been executed. To reflect this interdependency in the later BPA mapping we always put the sending nodes that reflect the sending branch before the receiving nodes that reflect the receiving branch of an AND-bond. The node(s) representing such a receiving branch are always put to the end of the sequence. This is depicted in the transformation of process p_1 in example 4 of Figure 48. This means the mapped intermediate throwing events in the later BPA always occur before the receiving events if they are on two different branches. In this case, our mapping is not trace equivalent anymore, but the trace that our BPA produces is contained in the traces that the process model level produces regarding the significant nodes. The interdependency between the process is retained in regard to initiator and strength of interdependency.

Example 2 in Figure 49 shows another example of an AND-bond aggregation. The later transformation of the AND-bond with in total five nodes results in a sequence of two events with multiplicities. The

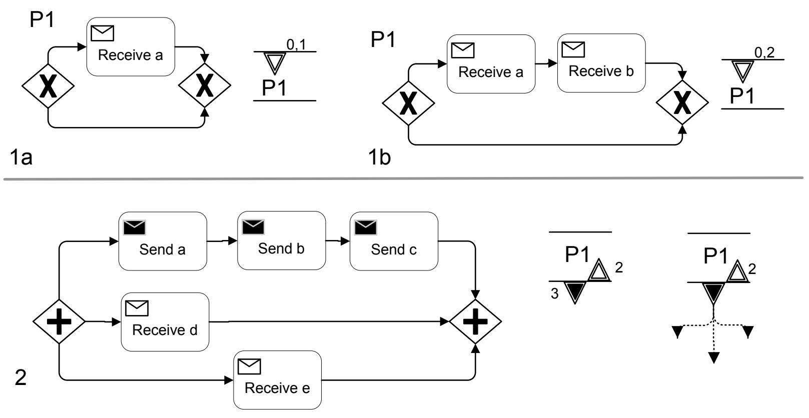


Figure 49: Exemplary XOR- and AND-bond aggregation and BPA transformation

receiving nodes occur after the sending nodes have occurred. In example 2 we see two different BPA transformations results of the process model construct in regard to flows and multiplicities assigned to the BPA events. If one or the other is chosen depends on the arrangement of nodes of the partner process(es). If there is only one partner process and all receiving nodes can be aggregated as well, the aggregation results in one sending event and one receiving event with $\mu(e) = \{3\}$ as also depicted in example 3 of Figure 48. If the receiving nodes cannot be aggregated or belong to different partners, the sending of messages is represented by the same amount of flows as also depicted in example 2 in Figure 48.

An AND-bond can also be mapped to an according node sequence if at the utmost one of the branches consists of mixed nodes (sending, receiving). In this case, the aggregation of the nodes of the sending branch are put first, then the aggregation of the mixed branch, and at last the aggregation of nodes of the receiving branch.

The AND-bond aggregation into node sequences becomes difficult if the polygons on more than one branch contain nodes of different types, i. e., sending and receiving nodes. In this case, the AND-bond cannot be mapped to an according satisfying sequence in the later BPA process without adding restrictions to the BPA that do not exist on process model level, e. g., introducing deadlocks in the worst case. AND-bonds with several branches with mixed node types cannot be extracted and are excluded from our extraction algorithm as depicted in Figure 50.

AGGREGATION OF XOR-BONDS. Similar to the aggregation of AND-bonds, each branch of an *XOR-bond* component is aggregated , analyzed, and compared to the other branches. The comparison results in different situations. The aggregation of XOR-bonds is restricted by the fact that the whole bond has to be aggregated to only one node. The aggregation of an XOR-bond to a sequence of nodes would in-

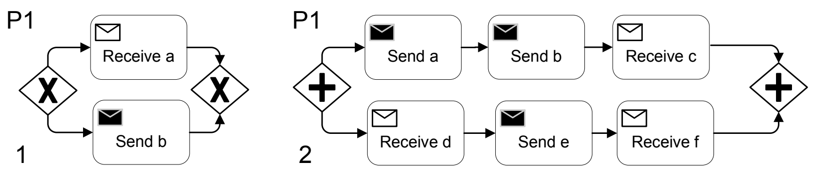


Figure 50: Examples of not covered BPA transformations

troduce more behavior into the later BPA as the behavior of several BPA events in a sequence that are part of conflicting flows is depicted by all possible combinations of event occurrences. Our BPA concept provides no mechanism to depict a sequence of conflicting flows in one element. In this regard our extraction algorithm is limited. In future work, BPAs could be extended by new complex elements that allow for such representation. In the simplest situation each branch of an XOR-bond contains of one node of the same node type and only the partner processes of each branch differ. In this case the nodes of the XOR-bond are aggregated to one node. The information that both nodes are in conflict is stored as well and kept for the later transformation into BPA constructs. This situation is depicted by process p3 in example 4 in Figure 48. The XOR-bond is mapped to one sending BPA event and the conflict relation is inserted into the BPA connecting the different recipients with the sender. The BPA conflict relation allows disembodying the exclusive behavior in a process model to inter process behavior.

Figure 49 shows examples of XOR-bond aggregations in which one branch has no node and the other either one or two nodes. Here, the nodes can be aggregated to one node as they are neighboring. The information that the sending or receiving is optional, depicted by the empty branch, is defined by the according elements in the multiplicity set of the BPA event. For instance, in our example 1a and 1b in Figure 49 the multiplicity of the BPA receiving event representing the XOR-bond of the according process model is set to $\mu(e) = \{0, 1\}$, respectively $\mu(e) = \{0, 2\}$.

Although, the BPA conflict relation allows to depict exclusive behavior of process models on BPA level, our algorithm is limited in such situations. Sequences larger than one node on an XOR bond branch lead to difficulties for the aggregation and later extraction to BPA events. If the nodes of each branch have the same partner process and are of the same type, we can aggregate them to one event as they depict the same sequence and have the same partner as depicted in examples 1a and 1b of Figure 49. However, XOR-bonds with only one partner per branch, the same number of nodes, and the same type of nodes for the whole XOR-bond are only aggregated if all partner nodes can be aggregated to one node as well.

The number of partners per branch cannot exceed 1 for the aggregation of XOR-bonds as such structure could not be extracted to an according BPA structure. XOR-bonds with mixed type of nodes cannot be aggregated and extracted with the current configuration of our algorithm, either. Such a component with a sending node on one branch and a receiving node on the other is shown in Figure 50.

Conceptually, it is possible to extract more complex XOR-bonds. There are different solutions that all come with drawbacks in clarity of the resulting BPA. For instance, the above mentioned XOR-bond could be mapped to a sequence of sending and receiving events with optional multiplicity. Each of these events would also be part of a conflict relation on BPA level. However, inserting them in a BPA process with the according conflict relations would add additional behavior and relations on BPA level. Hence, we refrain from such aggregations.

The *Loop-bond* is a special type of XOR-bond with two branches that either both contain a node or only one of them, similar to the examples in Figure 49. Both branches must be compared and if both branches contain the same type of nodes the *Loop-bond* is aggregated to one node. The information on the possible amount of iterations is described by the condition of the Loop-bond structure and is kept for defining the multiplicity set of the BPA event.

FROM AGGREGATED NODES TO BPA EVENTS AND FLOWS. The elimination and aggregation of significant nodes up to the first level of the RPST_{PM} results into a polygon of aggregated nodes. If desired, this polygon can be aggregated once more by aggregating neighboring nodes of same type. For the transformation of aggregated nodes to BPA events the polygon is traversed and for each node an according mapping to BPA events is determined. The leftmost element in the polygon is the start node and the rightmost is the end node. For those nodes the according start and end events are created in the BPA process. All other significant nodes are mapped to intermediary throwing or catching events in the BPA process.

In many cases the aggregated nodes will have postsets or presets with several entries. To determine their transformation to one or several BPA events, we consider the aggregated nodes of their partner processes. Depending on the different partners and the number of relations to each partner (entries in the post or preset), either one event with the according multiplicity elements, several flows, or several events are created for the according BPA processes. For instance, in example 2 of Figure 48 the aggregated node that represents the AND-bond is transformed into one BPA event with two flows whereas in example 3, the same AND-bond is transformed into one event with $\mu(e) = \{2\}$. This is because both message flows have the same partner process and point to the same aggregated node of the partner process.

If the number of flows between all partner processes differ, the transformation of an aggregated node may require the creation of several BPA events with different multiplicities that reflect the gathered interdependencies. If the stored information says that particular entries in the pre- or postset were in conflict, then the conflict relation is defined for the according flows of the BPA. Such a transformation can be seen in example 4 in Figure 48.

Our aggregation and extraction algorithm has two advantages, on the one hand it provides the ability to extract BPAs from process models of higher complexity covering a larger set of process models. On the other hand it provides a possibility to generate compact BPAs that are reduced to only core interdependencies. Depending on the magnitude of aggregation desired, the algorithm could be configured to generate compact or more fine granular BPAs.

Our ON analysis technique can be applied after extracting BPA according to the complex extraction algorithm, however only with restrictions. The complex BPA extraction algorithm is not trace equivalent but retains the causal interdependency between processes, i. e., that the trace depicted on BPA level may be restricted to particular execution traces but does not create problems that do not exist on process model level.

The only in-specificity is caused by the aggregation of several receiving nodes on one branch that have different partners. In these cases, the execution of several instances of one process could induce the occurrence of the receiving event although the messages sent originated from the same partner instead from both as depicted in the process model. Except of these cases described above, the analysis can be applied to BPAs extracted with the complex algorithm. This problem can be overcome by extending the verification formulae with additional information, e. g., checking if the according sending events of each process have occurred.

Listing 1, Listing 2, and Listing 3 show parts of the algorithm for elimination and aggregation in simplified form as pseudo code. It recursively traverses the tree. The elimination and aggregation of nodes is performed in the same traversal of the tree. It deletes insignificant nodes and collects the significant nodes that are then aggregated. Each component is analyzed, resolved, and an aggregation is determined. The strategy for resolving each component is determined by its type, being a polygon, XOR-bond or AND-bond. The transformation of the aggregated nodes to BPA events and processes is omitted.

Listing 1: Pseudo code complex extraction algorithm

```
Input: Set of interdependent Process Models
Output: BPA

Function Main(List<ProcessModel> pms){
        BPA bpa = new BPA();
        bpa = createBPA(pms);
}
```

```
BPA createBPA(List<ProcessModel> pms){

        for(pm in pms){
                RPST rpst = constructRPST(pm);
                List<ComplexList<events>> bpaseq = parseTree(rpst
                    .getRoot());
                bpa.addProcess(createBPAProcess(bpaseq));
        }
        bpa.drawRelation;
        return bpa;
}

List<ComplexList<events>> parseTree(Node rootnode){
        List<Node> children = rootnode.getChildren();
        List<ComplexList<events>> seq = new List<ComplexList<
            events>>;
                for(node in children){
                        if(node.equals(‘‘Leaf’’) && node.
                            isSignificant()){
                                seq.add(node);
                        }       else{
                                        seq.add(parseTree(node));
                        }
                }
                switch(rootnode.getType)(){
                        case XOR-Bond :  handleComponent(‘‘XOR’’,
                            seq);
                        break;
                        case Loop-Bond : handleComponent(‘‘Loop’’
                            ,seq);
                        break;
                        case AND-Bond : handleComponent(‘‘AND’’,
                            seq);
                        break;
                        case Polygon :
                            handleComponent(‘‘Polygon’’,seq);
                        break;
                }
                return seq;
}
```

Listing 2: Pseudo code handle component

```
List<ComplexList<events>> handleComponent(String nodetype, List<
    ComplexList<events>> sequence){
If(nodetype.equals(‘‘XOR’’ || ‘‘AND’’ || ‘‘Loop’’ )){
                                determineAggregation(sequence,
                                    nodetype);
} else if(nodetype.equals(‘‘Polygon’’)){
  sequence =    aggregatePolygon(sequence);
}
return sequence;
}
```

Listing 3: Pseudo code aggregate polygon

```
List<ComplexList<events>> aggregatePolygon(List<ComplexList<
    events>> sequence){
        for(sequence.iterator; iterator.hasNext){
                        if(element.getType() == sequence.next().
                            getType){
                                        element.addPartners(
                                            sequence.next().
                                            getPartners());
                                        sequence.next().delete;
                        }
                }
        return sequence;
}
```

7.2.6 *Extended Extraction Approach*

Based on our initial basic extraction approach described in [39] and in [43] another extended BPA extraction approach was taken by Breske in his master thesis [15]. He extended our BPA concept with an arbitrary split and an arbitrary join event structure to be able to extract BPAs from process models of higher complexity than described in Section 7.2. The behavior of the arbitrary split can be described as a combination of our multicast net and a send-conflict net, such that the multicast net first produces the amount of tokens specified by the event's multiplicity set and then distributes them arbitrarily to the processes connected through the conflict net. The behavior of the arbitrary join is similar to our collector net and allows a process to be instantiated by a subset of incoming triggers specified by the event's multiplicity. In this way, any kind of firing sequence can be produced.

To classify the behavior encountered in the RPST_{PM} fragments, Breske defined behavioral categories. All complex behavior that could not be described in a simple way, was mapped to the complex category. Those categories were again mapped to the according sequence of events on BPA level and the observation of complex behavior was mapped to the arbitrary split or arbitrary join structure. In this regard, he accepted to loose specificity of BPAs but enabled the extraction of mixed sequences in AND- and XOR-bonds by mapping them to so called complex blocks that define more behavior of the process interaction than depicted on the process model level. His dictum is to allow more behavior in BPAs but never less behavior than defined on process model level.

In his work he also provided an implementation of the extraction algorithm based on the *jBPT*[1] format and integrated it into the PromniCAT platform[2], a tool developed under our supervision to perform analysis on large process model collections [41]. jBPT is a Java-library that leverages graph structures to support a canonical process representation providing descendants for each supported modeling language. The implementation of his BPA extraction tool is described as example use of the PromniCAT platform in Section 9.2.1.7.

7.3 FROM BPA TO BUSINESS PROCESS MODELS

The transformation from BPA to process models is used when using our BPA approach top-down. Using the newly modeled BPA as input, the transformation algorithm generates a process model skeleton on process model level for each business process depicted in the BPA. For each event of the BPA process the according event is created on process model level, e. g., for a BPA start event a plain BPMN start event is created in the process pool. If that event takes part in a trigger or information flow relation, then the event is typed into an according receiving or sending event, e. g., a BPMN start message event. Alternatively, if the event is an intermediate event on BPA level, the event could also be transformed into a sending or receiving task on process model level. In general we resort to the transformation from BPA events to process model level events. The transformation begins with the start event of a BPA process and then continues the sequence of intermediate events until it reaches the end event. Business Process Architectures processes are sequences. Hence, the order of the events of BPA processes is reproduced on the process model level by connecting the events with a sequence flow.

After all BPA processes have been transformed into process model skeletons on process model level, the trigger and information flow relations need to be transferred to process model level. This step depends on the underlying process modeling notation and process modeling tool and has to be implemented accordingly. In BPMN one can transfer these relations into message flow relations depicted by a dotted arc. Each relation on BPA level is translated by connecting the according events on process model level. For instance a trigger relation is transformed in such a way that the message end event of the triggering process is connected by a message flow with the message start event of the triggered process. Such message flow may be attached to a collapsed pool that acts as proxy for the partner process. The exact information of trigger and information flow relations may be hidden in the attributes of the events in the process modeling tool, e. g., as URI of the target or source event. The concrete transformation

1 http://code.google.com/p/jbpt/
2 http://code.google.com/p/promnicat

depends on the capabilities of the process modeling tool and the underlying modeling guidelines a company has adopted to comply to. Figure 51 shows an exemplary transformation from BPA process (Figure 51a) to a process model skeleton (Figure 51b) and subsequently to an enriched and more detailed process model (Figure 51c).

We define the transformation from BPA to process models in two steps. First we transform all BPA business processes into process model skeletons on process model level. In a second step we add the information flow and trigger relations. To assure consistency between both levels we define consistency criteria that the process models on process model level need to comply to. The transformation from BPA processes to business process model skeleton is defined as follows.

Definition 47 (BPA Process to PM Skeleton Transformation) *Let* $BPA = (E, V, L, I, \mu, \chi, \pm)$ *be a business process architecture. Let* $PMC = (\mathcal{PM}, MF)$ *be the according process model collection with* $\mathcal{PM}$ *being a set of process model skeletons. Let* $v = \langle e_1, e_2, \dots, e_k \rangle$ *be a* BPA *process. Let* $PMS = (N, CF, type, s, f) \in \mathcal{PM}$ *be a process model skeleton.* $N \subseteq A \cup E \cup G$ *depicts the set of nodes with* A *being the set of activities,* E *the set of Events, and* G *the set of gateways of the process model skeleton* PMS. *There is a bijective mapping between* $v \in BPA$ *and* $PMS \in \mathcal{PM}$*, i.e., for every process in the* BPA *there exists exactly one process model in the* PMC. *The transformation from a BPA process to a process model skeleton is defined as follows:*

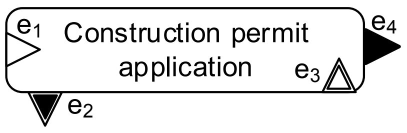
- $N = \{n_i | e_i \in v\}$
- $\forall 1 \leq i \leq k (e_i, e_{i+1}) \in v \Leftrightarrow (n_i, n_{i+1}) \in CF$
- $s = n_1, e = n_k$ *depict the start and end node of the process model skeleton*
- $CF \subseteq N \times N$*, such that* $\forall (x, y) \in CF : \exists e_i, e_{i+1} \in V, 1 \leq i < n, \wedge x = e_i \wedge y = e_{i+1}$ *and* $\forall n \in N \setminus \{s, f\} : |\bullet n| = |n \bullet| = 1 \wedge (n, n) \neq CF$
- $mult(n) = \mu(e)$ *denotes that the multiplicity of process node is equal to the execution times according to the* BPA*'s event* ◇

The result of the first transformation step is a set of process model skeletons that contains the same amount of processes as depicted in the BPA. To finish the complete transformation the according information flow and trigger relation of the BPA have to be mapped to the process model skeletons. This is done by representing them in according message flows that connect sending nodes from one process model to the receiving nodes of another process model. For each event pair in an information flow or trigger relation the according nodes in the process models are connected by a message flow. The first member of the event pair maps to the source node and the second member of the event pair maps to the target node. The transformation of the information flow and trigger relations is defined as follows.

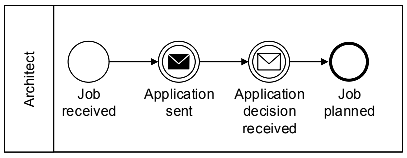
Definition 48 (BPA Relation to Message Flow Transformation) *Let* $BPA = (E, V, L, I, \mu, \chi, \pm)$ *be a business process architecture. Let* $PMC = (\mathcal{PM}, MF)$ *be the according process model collection with* $\mathcal{PM}$ *being a set of process model skeletons. Let* $PMS = (N, CF, type, s, f)$ *and* $PMS' = (N, CF, type, s, f)$ *be a pair of process model skeletons in* $\mathcal{PM}$*. The transformation from BPA information flow and trigger relations to process model message flows is defined as follows, such that for every pair* PMS, PMS' *holds:*

- $MF = \{(n_i, n_j) \in (N \times N') \cup (N' \times N) | (e_i, e_j) \in I \wedge \bullet n_j = \emptyset\} \cup$ $\{(n_i, n_j) \in (N \times N') \cup (N' \times N) | (e_i, e_j) \in L \wedge \bullet n_j \neq \emptyset \wedge n_j \bullet \neq \emptyset\}$ ◇

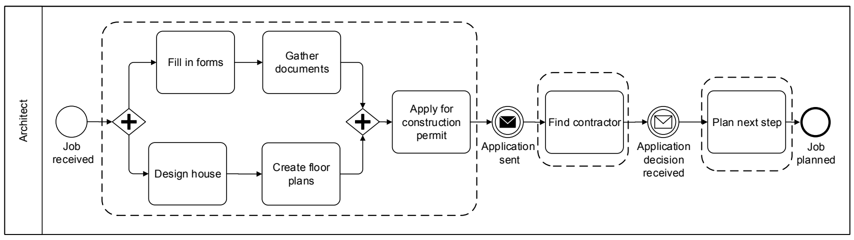
The generated process model skeletons can be enriched by adding activities, events, and other workflow elements. While elaborating the business process model skeleton to a complete process model, the basic skeleton structure provided by the BPA transformation has to be kept. This means that only new control flow elements that do not interact with another process can be inserted between the skeleton elements. For instance, the insertion of new sending and receiving nodes would result into a breach of the sequence provided by the BPA process.



(a) BPA process



(b) Process model skeleton



(c) Enrich process model

Figure 51: BPA process to process model transformation

Considering the techniques for process model decomposition the resulting process model skeletons depict one polygon with trivial components. The process model skeletons can be enriched with very complex control flow structures by inserting bond components of different depth between two trivial components under the precondition that they do not contain any sending or receiving node. Figure 51 shows the transformation of the "construction permit application" process into a process model skeleton and its further enrichment to a detailed process model. The events of the BPA process are transformed to BPMN events. The start and end event are transformed into blank events as they do not take part in a trigger or information flow relation. Figure 51b and Figure 51c illustrate the refinement step from process model skeleton to an enriched process model, e. g., an AND-bond component and a polygon were inserted between start and throwing intermediate event. A simple task was added between intermediate throwing and intermediate catching as well as between the intermediate catching and the end event. The nodes on process model level that represent events of the according BPA process will occur during each process execution on process model level. Using the concepts from Section 7.2 on BPA extraction, the sending and receiving nodes generated by the process model skeleton transformation could also be replaced by bond components that reflect the same behavior.

Consistency between both levels of abstraction is of utmost importance. Section 7.2 defined consistency criteria for a bottom-up BPA extraction approach. Similarly, we define consistency criteria for the top-down approach to assure consistency between the BPA and the process model level.

Definition 49 (Top-down Transformation Consistency Criteria) *The generation of process models from BPAs must comply to the following consistency criteria:*

- *For each BPA process a process model on process model level must exist*
- *The process model must retain the order of events depicted by the BPA process*
- *For each event in a BPA process exists a matching process model component in the according process model*
- *For each trigger and information flow of the BPA a matching message flow on process model level exists* ⋄

The consistency criteria demands that for all business processes contained in the BPA there exists one process model in the process model collection. For each sending and receiving event in a BPA process, there is one sending node, respectively receiving node, in the according process on process model level. The order provided by the events on BPA level must be retained by the matching sending and receiving nodes in the process model. We assure this by allowing only the insertion of process model components between two skeleton nodes that do not contain any sending or receiving event. With the help of these consistency criteria, the transformation from BPA to process model level and the further elaboration of the process models stays consistent.

7.4 SUMMARY

This chapter introduced a novel application methodology for our BPA approach that describes its use top-down and bottom-up, i. e., the steps to perform to create a consistent process model collection structure beginning from different starting situations. PA approaches presented in Section 3.2 can be applied either top-down or bottom-up. Only very few approaches describe concrete guidelines to create a PA, e. g., Dijkman et al. [31, 35] and zur Muehlen [106]. In this regard, we contributed a novel approach that provides concrete techniques and steps for the creation of PAs.

The methodology encompasses algorithms that specify the BPA extraction from process models as well as the process model generation from BPA processes. The BPA extraction from process models uses and extends existing business process model abstraction algorithms based on the $\mathsf{RPST}_{\mathsf{PM}}$ [136, 117, 160] by including business process interdependencies in the abstraction process. Considering business process interdependencies complicates the abstraction of process models as the granularity of extracted elements depends on the characteristics of the partner processes. Existing business process model abstraction approaches focus on the abstraction of single process models only (see Section 3.4). They introduce a range of techniques to aggregate, eliminate, and hide nodes or elements of a business process, e. g., Eshuis et al. [47] and Reichert et al. [121] provide customized views by hiding not relevant nodes and highlighting elements of interests like activities to be performed by a user or activities that are relevant for a customer. Reijers et al. [122] take a step back and empirically investigate the effects of modularization in process models by using sub-processes for the process model understanding. Their findings show that using sub-processes has positive influence on the process model understanding. Our approach reduces processes to only few elements and highlights the interdependencies between processes in form of trigger and information flow relations. It goes beyond process model borders and connects them to large scenarios of interdependent processes showing only important elements in this context. This is the starting point for analyzing processes in their context, e. g., by enriching them with additional meta-data.

The top-down decomposition of BPAs to process models is similar to the public-to-private (p2p) approach described by Aalst and Weske [150]. Both approaches aim at the construction of correct process interaction respectively interdependencies by design. The p2p approach demands the structural properties of a Workflow net for the description of the inter-organizational interaction. The top down use of our BPA approach and the p2p approach differ in terms of granularity level and scope. Whereas the p2p approach describes the inter-organizational message exchange in detail on process model level, our BPA approach provides a high level end-to-end abstraction of processes within an organization. Business process choreography diagrams as well, focus on the message interaction between processes but resort to an actor view, such that the involved processes are hidden, and only the interaction between actors of different organizations is described.

Our BPA methodology is the first that encompasses BPA analysis, BPA extraction, and process model generation techniques that support the consistent modeling of business processes and their interdependencies throughout different levels of a process model repository. It facilitates the maintenance and improvement of the quality of process model collections by harmonizing the process relations and achieving consistency between process presentations on different levels. Chapter 8 extends our BPA approach to also incorporate data aspects in the extraction of BPAs and allows visualizing data interdependencies between process models in a Business Process Data Architecture.

8 DATA ASPECTS IN BUSINESS PROCESS ARCHITECTURES

This chapter is based on the published papers "Deriving Business Process Data Architectures from Process Model Collections" [43] and "From Process Models to Business Process Architectures: Connecting the Layers" [39]. It deals with data aspects in process models for creating Business Process Data Architectures and shows how to identify process interdependencies based on data modifications of activities. It extends the BPA extraction algorithm with data aspects.

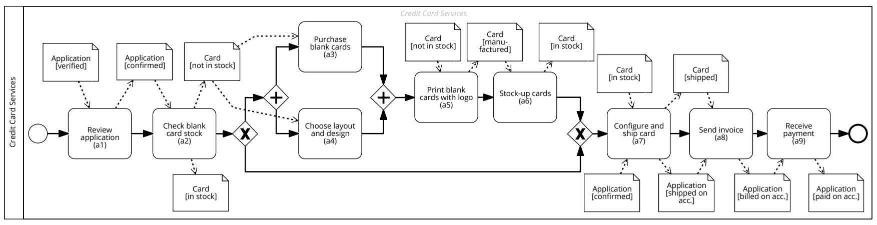
Data aspects have gained importance in business process modeling. Most work in this field has been carried out in regard to modeling data aspects, examine consistency issues between the data model and control flow, or synchronizing different OLCs of a process model along the research streams of activity centric and object centric process modeling [48, 77, 103].

As more and more process models depict the use of data objects, their interdependencies need to be considered and examined when creating BPAs. In the previous chapter we introduced a BPA extraction algorithm that creates a BPA from a given process model collection. In this chapter we focus on the extraction of data interdependencies between business processes which has not been discussed in Section 7.2. In many cases the data interdependencies prevailing between business processes are hidden as the exchange of data objects is not modeled explicitly and rather happens indirect through access to shared information systems.

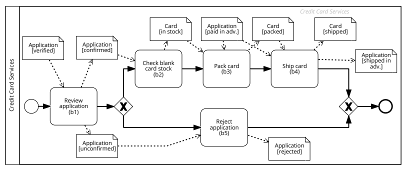
Nevertheless, process interrelations can be deduced by looking at the data objects and how processes manipulate them. For example, two processes have to be carried out in sequence, if one process produces a certain data object that another process consumes.

In the following, we introduce an exemplary scenario, describe different types of data interdependencies that can be observed between business processes, show how to create a *process data dependency matrix (PDM)* and describe an extraction algorithm to construct a *Business Process Data Architecture (data BPA)*, which reveals and depicts hidden data-related interdependencies and thus eases the management of process model collections. The resulting data BPA can be verified with the method presented in Chapter 6 unveiling erroneous process interaction due to data.

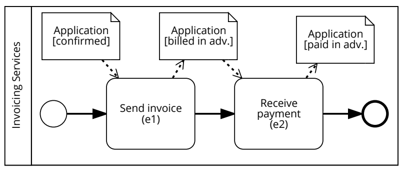
For the extraction of data interdependencies we refer to the process model definition from Definition 1, the definition of data objects and states from Definition 2, and the OLC definition from Definition 3.



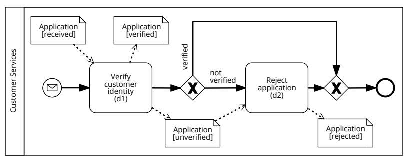
(a) Process p1: Credit card application process



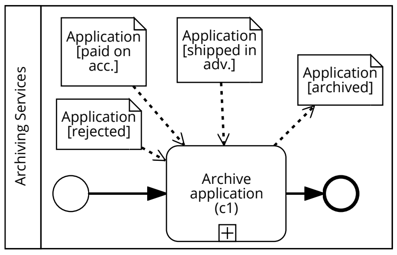
(b) Process p2: Credit card application process (adv. payment)



(c) Process p5: Invoice handling



(d) Process p4: Customer identity verification



(e) Process p3: Archiving

Figure 52: Customer identity verification and invoice handling process models

We assume that the underlying process models are structurally sound and block structured. Furthermore, we assume that all process models satisfy the notion of weak conformance [102] with respect to the utilized data objects, i. e., process models and OLCs do not contradict. We assume that all activities' data object accesses are modeled and that no activity writes a data object it has not read before and that each data state transition is realized by at least one activity from a process model. In regard to OLCs, we consider only *acyclic* OLCs and that there exists one shared object life cycle for all business processes using the same data objects.

8.1 DATA SCENARIO

We use the scenario from the processing of credit card applications at a financial service provider introduced in Section 1.2.1. We extend it to five processes and add data annotations. The scenario consist of the credit card application process in Figure 52a, a second alternative of the credit card application process with advance payment in Figure 52b, the customer identification process in Figure 52e, the invoicing process in Figure 52d, and the archiving process in Figure 52c.

Each credit card application that is received is handled by the customer verification process in Figure 52d. If the customer provided all required information, the credit card application can be handled, otherwise the application is rejected.

Process p_1 in Figure 52a describes the standard procedure for credit card applications. If a verified application arrives, an employee reviews the application and then checks if there are still blank cards in stock for issuing the credit card. If they are still in stock then the credit card is configured, the invoice is sent, and the payment is received. If the stock is empty the employee orders new blank cards and chooses a layout and design, and the stock of blank card is refilled. Process p_2 Figure 52b describes the credit card application process where a first initial deposit is required. It also allows to reject a credit card application after the reviewing of the application, e. g., when the employees re-classifies the customer as blacklisted in regard to the data provided.

The service of providing a credit card needs to be paid by the customers which is handled by the invoice handling process depicted in Figure 52c. At last, all credit card applications are archived by the archiving process in Figure 52e.

These five processes share the two data objects "application" and "credit card". Each data object has an OLC that describes the state transformation allowed on the data object. The data object life cycles are depicted in Figure 53 and Figure 54 in Section 8.2.

8.2 ANNOTATING THE OBJECT LIFE CYCLE

The first step to determine data interdependencies between the business processes is to annotate the OLCs of the data objects "application" and "credit card" with the activities that induce state transitions in the OLC. Activities in process models can either have reading or modifying data access. Read access is depicted by a dashed arc from the data object to an activity in a process and the modifying access by a dashed arc from the activity to the data object. A modifying access changes the state of the data object corresponding to a state transition in the OLC. As the data objects are shared and hence state transitions can be induced by activities from different processes, we label the state transitions with pairs of process and activity. For this step we define the relation $\mathit{map} \subseteq (S \times \mathcal{PM} \times \mathcal{A} \times S)$, where $\mathcal{PM}$ and $\mathcal{A}$ are sets of process models and activities. The *map* relation is visualized as arc inscriptions in the OLC, e. g., the modifying access is reflected by the inscription $p_2[b_3]$ on the arc between states "in stock" and "packed" in Figure 53. A reading access of an activity without transforming a data object is visualized as a dashed arc pointing away from the state that is read. E. g., the reading of data object state "not in stock" of the data object "credit card" by activity "purchase blank cards (a3)" of process p_1 is depicted as inscription $p_1[a_3]$ in Figure 53. In the annotated OLC this is reflected by $map(S, S) = (p, a)$, where S is the data state to be read, a is the activity, which performs the reading access, and p is the process of a. In the following we refer to the activities by the abbreviations in parenthesis written in the label, e. g., activity "a3" refers to the activity "purchase blank cards (a3)" of process p_1.

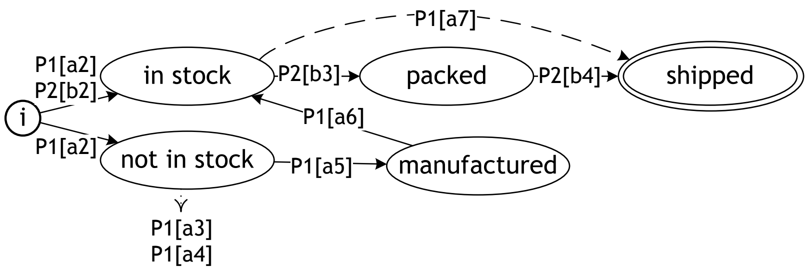


Figure 53: Annotated credit card object life cycle

Activities can perform modifications that induce multiple state transitions in the OLC along a path in the OLC and are not limited to directly succeeding states. Such a transition is depicted by an additional dashed arc leading from the originating state to the source state in the OLC, e. g., in Figure 53 the states "in stock" and "shipped" are connected by a dashed arc. In this case activity $p_1[a_7]$ induces a state transition from "in stock" and "shipped" omitting the state "packed" in the OLC.

8.3 DERIVING THE PROCESS DATA RELATION MATRIX

To determine the type of data interdependencies between processes, we first identify data interdependencies between their activities, and aggregate all found data interdependencies for a pair of processes in a data relation type. The data relations for all processes are then summarized in a process data dependency matrix.

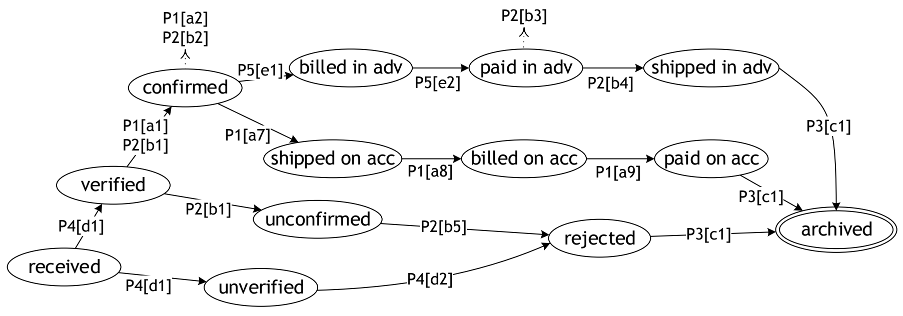


Figure 54: Annotated application object life cycle

8.3.1 *Deriving Direct Data Interdependencis from the Data Object Life Cycle*

Using the annotated life cycle we can derive direct data interdependencies between activities and consequently data interdependencies between business processes. Two succeeding state transitions in the OLC that are induced by activities from different processes describe an interdependency between these processes.

Depending on the amount of preceding and succeeding states in the OLC and the origin of activities inducing state transitions different relations between the processes can be derived from the OLC. In the following we describe the different relation types and how to determine them from the OLC.

Two activities from two different processes are dependent ($p[a_i] -\!\blacksquare\!\!\rightarrow p'[a_j]$) if one activity $p'[a_j]$ reads or modifies a data object that another activity $p[a_i]$ has modified before. For example the transition "unverified" $\xrightarrow{p_4[d_2]}$ "rejected" of data object "application" is followed by the transition "rejected" $\xrightarrow{p_3[c_1]}$ "archived". Because c_1 modifies a state written by d_2 it can only occur afterwards. The relation $-\!\blacksquare\!\!\rightarrow$ denotes direct data dependency between two activities and their processes.

A data object state transition can be induced by only one activity at a time. If several activities from different processes can perform a data state transition, they are in conflict with each other. This conflict needs to be considered when extracting data dependencies between processes.

In the OLC these conflicts are determined by traversing the OLC and checking for each state its successor and predecessor relations. Conflicts derive from two different situations observed in the OLC.

The first situation is observed when two different activities induce the same state transition. In the OLC this is visualized by an arc with two inscriptions, e. g., the state transition from the state "verified" to "confirmed" in the OLC of data object "application" can be induced by $p_1[a_1]$ and by $p_2[b_1]$ as depicted in Figure 54. In this conflict relation the state transition originates from the same state and has the same target state but can be realized by two different activities.

The second situation is observed if a state in the OLC has several preceding or succeeding states, and the state transitions leading to the state or leaving the state respectively are induced by activities of different processes, e. g., in the OLC of "application" both $p_5[e_1]$ and $p_1[a_7]$ originate in state "conflict" but induce the states "billed in adv." and "shipped on acc." respectively, and hence are in conflict. Similarly, the state "rejected" of data object "application" has two predecessor states "unconfirmed" and "unverified" and either $p_2[b_5]$ or $p_4[d_2]$ induce the state transition to the state "rejected" as shown in Figure 54.

As we are interested in interdependencies between processes we only consider those conflicts in which activities of different processes are involved. We differentiate between these conflicts as they require different handling when creating data BPAs. For this we refer to the first conflict situation described as same state conflict and to the second conflict situation as different state conflict.

To express these conflicts we define the conflict relation $\otimes$. Two annotated state transitions are in conflict, when they originate in the same source state or lead to the same target state and are induced by activities of different processes. We assume that conflicts in which only activities of one process are involved are taken care of by the control flow of the processes, i. e., we assume that there are no data related deadlocks in the process models.

To determine the relation between two processes the base relations $\mathrel{-\!\blacksquare\!\!\rightarrow}$ and $\otimes$ on activity level are combined and lifted to process level. To achieve this, all direct relations between the activities of two processes need to be considered along all paths of the OLC of a shared data object as different paths might show contradicting relations. The aggregation of relations to process level will be discussed in detail in the next section.

8.3.2 *Process Data Relations*

To determine the relation type between a pair of processes the OLC is traversed along each OLC paths and every base data relation is collected. Depending on the sequences of the different base relations observed for each path in the OLC the overall relation for the process pair is determined.

SEQUENTIAL DATA DEPENDENCY. Process p' *sequentially* depends on process p in regard to particular data object D (written as $p \rightarrow_D p'$), if all data accesses of process p happen before process p' accesses D for the first time on any path of the corresponding OLC. Additionally, there must exist at least one direct data dependency $p[x] \mathrel{-\!\blacksquare\!\!\rightarrow} p'[y]$. For example process p_3 sequentially depends on process p_4 in regard to the data object "application" as all data accesses of p_4 happen before the first data access of p_3 and there is a direct data dependency $p_4[d_2] \mathrel{-\!\blacksquare\!\!\rightarrow} p_3[c1]$ as shown in Figure 53.

We define two more variants of the sequential relation. Two processes p and p' are *sequentially overlapping* ($\xrightarrow{\uparrow}$) if $p \rightarrow p'$ and p additionally reads the last state it has modified in parallel with process p' on handover. Usually, this happens when a process hands over some work to another process but still uses the last data object state as input to continue its own work on another data object. E. g., in Figure 54, the relation $p_1 \xrightarrow{\uparrow} p_5$ is depicted on state "confirmed" where $p_1[a_2]$ reads the state that is also input to process p_5 that induces the next state transition to "billed in adv." with $p_5[e1]$. Hence p_1 and p_5 are sequentially overlapping.

A process p' *follows* ($\overset{\uparrow\uparrow\uparrow}{\rightarrow}$) a process p, if p' only reads the modifications performed on a data object by process p, i. e., if there are activities $x_1, \ldots, x_k$ of p and $y_1, \ldots, y_l$ of p' such that $p[x_i] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p'[y_j]$ for some $1 \leq i \leq k, 1 \leq j \leq l$ and each y_j has only reading access, e. g., the tracking of a delivery would result into such a relation.

EXCLUSIVE DATA DEPENDENCY. A process pair is exclusive if we observe one conflict relation between their activities in the according OLC. For example in Figure 54 we observe a conflict relation $p_1[a_1] \otimes p_2[b_1]$ as both activities originate in the same state and induce the state transition from "verified" to "confirmed" of data object "application". The exclusive relation is dominant, in the sense that it overrules other relations in ambiguous situations. E. g., on one path of the OLC for a data object D we observe a sequential data relation and on another path of the OLC we observe an exclusive relation for a pair of processes, then we classify both processes as being exclusive to each other.

Two processes p and p' are called *completely exclusive* ($p[\#]p'$) if we observe a conflict relation between their activities on all paths of the corresponding OLC, i. e.,$\exists p[x_i] \otimes p'[y_j]$ for all paths of the OLC.

INTERACTING PROCESSES. Two processes p, p' are *interacting* ($\rightsquigarrow$) if we observe on some path of the OLC direct data dependencies between activities of p and p' and vice versa, i. e., $p[x_i] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p'[y_j]$ and $p'[y_k] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p[x_l]$ for some activities x_i, x_l of p and y_j, y_k of p'. Additionally $x_i < x_l$ and $y_j \leq y_k$ have to hold regarding the behavioral profiles of p and p'. Because process p induces the first data dependency on the path, it is the initiator of the interaction. If the condition for interacting processes holds on different paths of an OLC the initiator has to be the same process on each path. Otherwise the processes are considered *contradicting* ($\perp$). Generally, interacting means that two processes take turns operating on a data object.

In the OLC of data object "application", we find direct data dependencies $p_2[b_1] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p_5[e_1]$ and $p_5[e_2] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p_2[b_3]$ (on the top path "verified"–"confirmed"–"billed in advance"–"paid in advance"–"shipped in advance"), and $b_1 < b_3$ as well as $e_1 < e_2$ hold. Therefore we can deduce that $p_2 \rightsquigarrow p_5$.

CONTRADICTING PROCESSES. Two processes are considered *contradicting* ($\perp$), if the relation between the two processes p and p' appears to be interacting, however both processes are initiators of the interaction on different paths of the OLC. For example, in one path we observe $p[x_i] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p'[y_n]$ and $p'[y_m] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p[x_j]$ and on another path we observe $p'[y_i] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p[x_n]$ and $p[x_m] \rightarrow\!\!\!\blacksquare\!\!\!\rightarrow p'[y_j]$.

These relations are the building blocks for creating a PDM for data interdependent processes.

8.3.3 *Aggregation of Multiple Data Object Relations in a Process Data Relation Matrix*

Business processes usually use more than one data object during process execution. Consequently, we need to consider the data relations found in all used data objects to determine the relation between a pair of processes. If we observe the same relation type between two processes on all shared data objects, then their overall relation type is defined by that type. However, the relations found in the OLC of the data objects may differ. For this case we define a dominance hierarchy between the relation types that can be applied if a link between two data objects can be established. Two data objects are linked if they are used by the same activities else they may be shared on different execution paths of both processes. In such ambiguous cases, process traces need to be considered which will be part of future work. If the data objects are dependent, i. e., are used by the same activities, we can resolve the ambiguity. Then the more dominant relation type determines the overall relation between the two processes.

If, on the one hand, the sequential relations for two processes p and p' agree in regard to all data objects they both access, the processes are said to be *sequentially* dependent ($\rightarrow$), so that for each data object $p \rightarrow p'$ holds. Similarly, this applies for sequentially overlapping and following relations.

If, on the other hand, the sequential relations differ for data objects D and D', e. g., $p \rightarrow_D p'$ and $p' \rightarrow_{D'} p$, then processes p, p' are *interacting*, except when their data accesses contradict. This is the case if each of the processes needs the other data object as input before inducing the state transition required by their partner process. Such a deadlock occurs, if $p[a] \mathrel{-\!\!\blacksquare\!\!\rightarrow} p'[b_1]$ in the OLC of D, $p'[b_2] \mathrel{-\!\!\blacksquare\!\!\rightarrow} p[a]$ in the OLC of D' and $b_1 < b_2$ in the behavioral profile of p' all hold.

Two processes p and p' are *exclusive*, $p\#p'$, if at least one relation on a data object between these processes is considered *exclusive* and the other relations on further data objects are considered *sequential*, *interacting*, or *following*.

We define the relation of two processes as *contradicting*, if we observe that they are contradicting for at least one data object. A PDM with contradicting processes will fail to terminate when turned into a data BPA because the contradiction would lead to deadlocks during execution. However, processes we identified as contradicting might succeed for some process traces if activities causing the contradiction are not executed in this trace. The identification of contradicting relations between processes indicates malformed data dependencies.

After having classified each process pair with a relation type the process relations are summarized in the process data dependency matrix. Table 5 shows the aggregated process relations for our "credit card application" scenario in the PDM and sets the coarse structure of the according data BPA. The later extracted data BPA must conform to the identified relations in the PDM.

Processes	p1	p2	p3	p4	p5
p1	–	#	$\rightarrow$	$\leftarrow$	#
p2	#	–	$\rightarrow$	$\leftarrow$	$\rightsquigarrow$
p3	$\leftarrow$	$\leftarrow$	–	$\leftarrow$	–
p4	$\rightarrow$	$\rightarrow$	$\rightarrow$	–	–
p5	#	$\leftsquigarrow$	–	–	–

(a) Process Data Relation Matrix (PDM)

Relation type	Symbol
sequential	$\leftarrow$ $\rightarrow$
sequ. overlapping	$\overset{\uparrow}{\leftarrow}$ $\overset{\uparrow}{\rightarrow}$
following	$\overset{\uparrow\uparrow\uparrow}{\leftarrow}$ $\overset{\uparrow\uparrow\uparrow}{\rightarrow}$
interacting	$\rightsquigarrow$ $\leftsquigarrow$
contradicting	$\perp$
exclusive	#
complete exclusive	[#]
no relation	–

(b) Relation type symbols

Table 5: Aggregated data object process relations for processes p1 to p5

As processes p_3, p_4, and p_5 only access the data object "application", their relations are determined by this data object. For example, process p_3 is *sequentially* dependent from process p_1, i. e., $p_1 \rightarrow p_3$ because of $p_1[a_9] \xrightarrow{\blacksquare} p_3[c_1]$ and no other relations exist.

Processes p_1, p_2 and p_3 sequentially depend on process p_4, process p_3 additionally is sequentially dependent on both p_1 and p_2.

Processes p_2 and p_5 *interact* with each other on data object "application" in the most upper path of the OLC. The interaction is initiated by process p_2 such that $p_2 \rightsquigarrow p_5$ holds as depicted in Table 5.

Processes p_1 and p_5 share only one data object. As we can observe an exclusive relation from the OLC of the data object "application" because of $p_1[a_7] \otimes p_5[e_1]$, we classify them as *exclusive*, such that $p_1 \# p_3$ on process level as well.

Processes p_1 and p_2 use both data objects, but because their relation is exclusive for both, we can unambiguously identify their overall relation as exclusive in the PDM as well. The remaining relations are determined accordingly; none is contradicting.

8.4 EXTRACTING THE DATA BPA

After having presented a technique to identify data dependencies between processes, we describe the extraction of a data BPA that provides an overview of the detected data interrelations. For the extraction of Business Process Data Architectures we require the process models, the annotated OLCs of the shared data objects, and the process relations that we observed for each data object.

We build on the extraction algorithm presented in Section 7.2. For the algorithm to be used with data objects in the process model, we map the data dependency observations made in the OLC to process model elements and complex control flow structures. The PDM (see Table 5) is used to determine the extraction strategy. It describes the overall structure of the data BPA, e. g., process precedence, or exclusivity.

8.4.1 *From Object Life Cycles and Data Interdependencies to Control Flow*

We introduce mappings from data dependency observations made in the OLC to known control flow structures that we then use for extracting a data BPA.

SEQUENTIAL DEPENDENCY. If we observe a direct data dependency between an activity $p[a_i] \mathrel{-\!\!\blacksquare\!\!\!\rightarrow} p'[a_j]$ in the OLC of a data object and the relation type between process p and p' is sequential, interacting, or following, we consider this direct data dependency a message flow from activity $p[a_i]$ to $p'[a_j]$. To depict the sending of information we insert a sending activity after the activity that induces the state transition in the source process model. In the target process model we insert a receiving activity before the activity that reads the data object.

EXCLUSIVE AND SAME STATE CONFLICT. If we observe a same state conflict relation between two activities in an OLC, such that $p[a_i] \otimes p'[a_j]$, we flag those activities insignificant for extraction and ignore these activities in the data BPA extraction process. Processes p and p' are in conflict as only either one can induce the state transition. In the PDM process p and p' are typed exclusive. Of course, the activities still need to be considered for extraction if they are in direct data dependency with activities of other processes.

EXCLUSIVE AND DIFFERENT STATE CONFLICT. If we observe a different state conflict relation between two activities in an OLC, we consider this an XOR-bond in the process model with an activity on each path that has a message flow to the according partner. E. g., we observe a different state conflict relation with $p[a_i] \mathrel{-\!\!\blacksquare\!\!\!\rightarrow} p'[a_j]$, $p[a_i] \mathrel{-\!\!\blacksquare\!\!\!\rightarrow} p''[a_k]$, and $p'[a_j] \otimes p''[a_k]$, then we treat those observations as XOR-bond with two paths where one path has an activity $p[a_{i1}]$ exhibiting a message flow to $p'[a_j]$, and the other path has an activity $p[a_{i2}]$ exhibiting a message flow to $p''[a_k]$. In the PDM process p' and p'' are typed exclusive. To represent the sending of

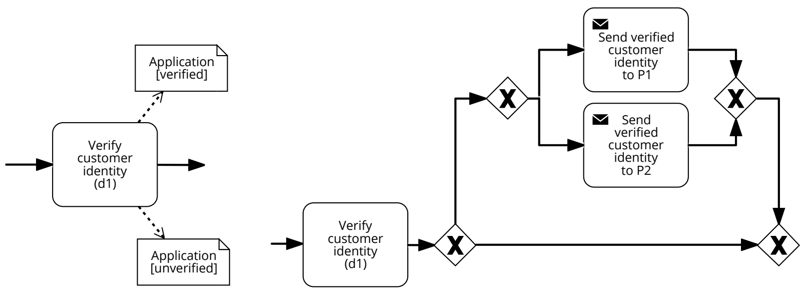

Application [verified]
Verify customer identity (d1)
Application [unverified]
Verify customer identity (d1)
Send verified customer identity to P1
Send verified customer identity to P2

Figure 55: Data write access to control flow transformation

information to one or the other partner the XOR-bond component is inserted after the activity. To represent the receiving of information from different partners the XOR-bond component is inserted before the according activity.

Figure 55 illustrates the complexity that is hidden in data objects of process models in regard to process interdependencies. Activity $p_4[d_1]$ induces either the state "verified" or "unverified" of data object "application". Hence, we introduce an XOR-bond component that represents this choice. To depict the choices we analyze the direct data dependencies of the data object for each state. Data object "application" in state "verified" can be read by either activity $p_1[a_1]$ of process p_1 or activity $p_2[b_1]$ of process p_2. Both processes are in conflict as only one activity can read the data object at a time. Hence, we introduce another XOR-bond as child component of the first XOR-bond in the process model with a sending task on each branch that either sends the message to the activity of process p_1 or the activity of process p_2. As the state "unverified" is read by an activity of the same process no sending task is introduced on the lower branch. The empty branch of the firstly introduced XOR-branch depicts that the sending of data object "application" in state "verified" is optional.

Figure 56 shows the receiving of information from different processes depicted by an input data object with different possible input states. Each data state is depicted by one branch with a receiving task in the XOR-bond. As the receiving of information is illustrated by the receiving tasks the XOR-bond is inserted before the activity.

Direct data dependencies between two processes that are typed exclusive in the PDM are ignored and will not be transformed into a trigger or information flow during BPA extraction.

CONTRADICTING. Processes being in a contradicting relation cannot be treated by our extraction algorithm and hence will not be considered. In many cases the contradicting relation hints at problems in the OLC and control flow of the process models as data or control flow deadlocks.

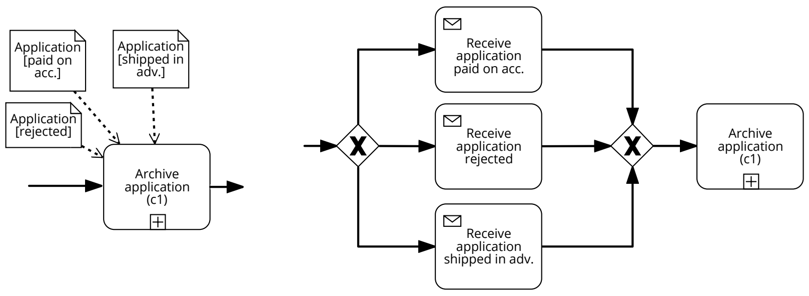

Application [paid on acc.]
Application [shipped in adv.]
Application [rejected]
Archive application (c1)
Receive application paid on acc.
Receive application rejected
Receive application shipped in adv.
Archive application (c1)

Figure 56: Data read access to control flow transformation

MULTIPLE DATA OBJECTS. An activity might have several data objects as its precondition/postcondition, e. g., activity $p_1[a]$ with object D_1 in state s_1 and data object E_1 in state t_1. For example, in our scenario we find such a situation with activity $p_2[b_4]$ that has data object "application" in state "shipped in adv." and data object "card" in state "shipped" as postcondition.

If additionally we observe direct data dependencies in the OLC of these data objects with $p_2[b_1]$ inducing the state transition of D_1 into state s_2 and $p_3[c_2]$ inducing the state transition of data object E_1 into state t_2, and p_2 and p_3 are not exclusive, such that $a \dashrightarrow b$ and $a \dashrightarrow c$, we transform such data construct into an AND-bond component in the process models with as many paths as partners observed for that activity. On each partner path we put the number of sending activities that equals the number of data objects the partner receives (for which there is a direct data dependency in the OLCs). In our example, we introduce an AND-bond component with two branches for activity $p_1[a]$. On one branch an activity $p_1[a_1]$ relates to $p_2[b_1]$ with message flow and on the other the activity $p_1[a_2]$ relates to $p_3[c_2]$ with a message flow. The BPA extraction algorithm transforms this construct into one sending event with two flows (trigger or information flow) to one receiving event of each partner process, illustrated in example 2 of the BPA extraction algorithm in Figure 48. If instead there is only one partner process and the receiving activities are in the same AND-bond component or neighboring in a polygon, the BPA extraction algorithm transform this construct into one sending event with multiplicity equals to the number of data objects, i. e., number of activities in the AND-block as depicted in example 3 of Figure 48.

8.4.2 *From Activities to Events.*

In Section 7.2 we presented a BPA extraction algorithm. As first step an $RPST_{PM}$ of each process is created. All nodes that do not take part in a message flow relation are eliminated. In regard to data objects, this means that all activities that are not part of a direct data dependency or different state conflict relation are eliminated. The remaining significant nodes are analyzed for their data dependency. Based on the data relation the according non-data control flow structures described in previous section are inserted into the process model. This step leaves us with a reduced $\mathrm{RPST}_{\mathrm{PM}}$ with only significant nodes, i. e., process nodes participating in a message flow (representing the data interdependencies).

We configure our BPA extraction algorithm to the data specific settings where we only consider activities. In this regard we fuse the start and end events with their neighboring activities. Hence, the leftmost (minimal) activity node in the direct children components of the root node of the $\mathrm{RPST}_{\mathrm{PM}}$ of the original model depicts the start node of the process and the rightmost (maximal) activity node the end node of the process. The first activity of the process model is mapped to a start event and the last activity to an end event in the BPA process. The BPA extraction algorithm consults the PDM when transforming the process model activities to BPA events. It ignores all connections between processes that are exclusive according to the PDM.

With this configuration we can traverse the $\mathrm{RPST}_{\mathrm{PM}}$ and identify a mapping to BPA processes and events according to the complex BPA extraction algorithm presented in Section 7.2.5. Depending on the data relation type, the $\mathrm{RPST}_{\mathrm{PM}}$ component type, the location of the activity, and number of partner processes, the extraction algorithm defines the according transformation to BPA event type and introduces the required amount of triggers and information flows into the resulting BPA. The BPA extraction algorithm places the mapped events in the according BPA process analog to their position in the $\mathrm{RPST}_{\mathrm{PM}}$. The order of events in the BPA process is determined by the position of the activity node in the $\mathrm{RPST}_{\mathrm{PM}}$. The result is a data BPA revealing the data interdependencies between business processes. There is a many-to-one mapping between activities and events because internal activities were removed and significant activities were aggregated to one event in the BPA.

Figure 57 depicts the resulting data BPA for our scenario. It consists of the five processes presented and their data dependencies. Figure 57 demonstrates how the conflict relation affects the creation of the data BPA. It is handled like an XOR-bond in a process model without any data objects. In Figure 54, activities $p_1[a_1]$ and $p_2[b_1]$ both transform data object "application" from state "verified" to "confirmed" and both are in direct data dependency with $p_4[d_1]$ because of $p_4[d_1] \xrightarrow{\blacksquare} p_1[a_1]$ and $p_4[d_1] \xrightarrow{\blacksquare} p_2[b_1]$. Hence, those activities are conflicting and the data BPA must prevent the processes, to which $\beta(a_1)$ and $\beta(b_1)$ belong, to be instantiated at the same time.

For the data BPA, this means that these trigger flows are in conflict, i. e., $(\beta(d_1), \beta(a_1)), (\beta(d_1), \beta(b_1)) \in \chi$ which is visualized by the XOR gateway in Figure 57. Activity $p_4[d_1]$ optionally triggers pro-

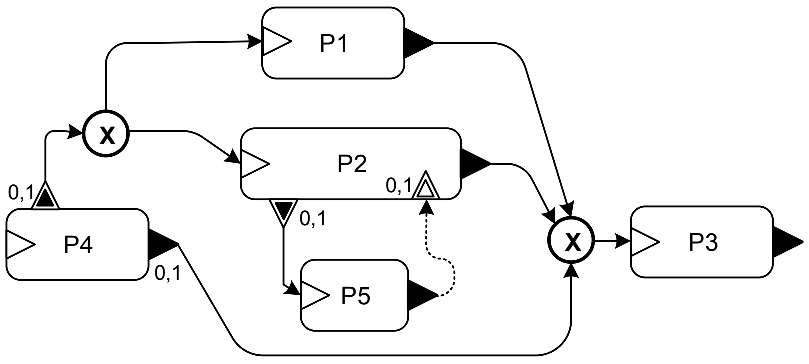


Figure 57: Resulting data BPA

cess p_1 or process p_2 which is depicted by the intermediate sending event with a multiplicity set of {0,1}. This due to activity $p_4[d_1]$ that has different data object states as postcondition. One of the states is in direct dependency with two different processes and the other state only refers to an internal activity.

Process p_4 optionally triggers process p_3 as the original model contains an XOR-bond that consists of a branch with a data interdependency and one empty branch. Consequently, the data interdependency only exists, if the according branch is taken in the source process model. This optionality is depicted in the BPA process by an end event with a multiplicity set of {0,1}.

Our data BPA process shows all behavior depicted in the process model although the data conditions restrict the process model execution to specific paths, e. g., a path were both events send a trigger cannot be executed on process model level due to the data object states.

Activity $p_2[b_1]$ modifies the state from "verified" to "confirmed", a state which is read by activity $p_5[e_1]$. Processes p_2 and p_5 have further data dependencies $p_2[b_1] \dashrightarrow p_5[e_1]$, $p_5[e_2] \dashrightarrow p_2[b_3]$ and $p_5[e_2] \dashrightarrow p_2[b_4]$, which each turn into a pair of a sending and a receiving event. For the sending intermediate and receiving intermediate events of process p_2 the correspondence relation is defined, such that both events have the same multiplicity set as depicted in Figure 57 and assures for a BPA run that they are equal.

Activity $p_2[b_4]$ is furthermore in data dependency with $p_3[c_1]$, meaning that $\beta(b_4)$ and $\beta(c_1)$ are in relation. As $p_1[a_9], p_2[b_5]$ and $p_4[d_2]$ are in conflict, they must be in conflict in the data BPA as well. Hence, the conflict relation symbol is introduced and connected to the start event of p_3 in the data BPA.

The extraction of a data BPA visualizes hidden data interdependencies between processes and enables analysis over several processes. Despite the complex data setting, the BPA extraction algorithm was adapted to handle data objects in process models. This was done by mapping the data modification patterns to known control flow structures reflecting their behavior. The easy integration of data aspects into our BPA concept demonstrated the flexibility of our BPA to be adapted to new domains. The examination of data interdependencies between business processes regarding contradicting relations of multiple data objects requires future research to determine the types and source of such contradictions and resolve them.

8.5 SUMMARY

In this chapter we introduced a novel technique to identify data interdependencies between processes based on data objects and their OLCs and extract them to Business Process Data Architectures. It maps the data interdependencies to common control flow constructs to resort to the BPA extraction algorithm from Section 7.2. Our research on data interdependencies between processes can be integrated into research on data aspects in process models and process choreographies (see also Section 3.3). Research on data in activity-centric process models deals with assuring consistency between the data and control flow perspectives [48, 77], e. g., by synchronization of several object livecycles (OLCs) of different data objects for one process model [103].

However, only few approaches exist that examine data aspects in regard to process interaction, e. g., Knuplesch et al. [69, 68] and Fahland et al. [51]. Knuplesch et al. [69, 68] enrich BPMN choreography diagrams with a data layer that assigns virtual data objects to message exchanges between participants of a collaboration. In this approach the data objects are used to determine the routing through the choreography control flow. Taking a different focus, Fahland et al. [51] introduce a technique based on proclets to check conformance between execution logs and artifact-centric processes.

Being aware of the importance of data aspects, our approach supports the detection, visualization and analysis of hidden data interdependencies not only in regard to message exchanges but also in regard to their trigger semantics. On PA level, data aspects have been mostly ignored except from the use of business objects as grouping elements for business processes.

Part IV

EVALUATION AND CONCLUSION

9 BUSINESS PROCESS ARCHITECTURE EVALUATION

Parts of this chapter are based on the published papers [41, 46]. After we introduced our BPA approach and its different techniques, this chapter presents an evaluation of our BPA approach to validate its usefulness and applicability, also in comparison with other PA approaches.

9.1 CONCEPTUAL EVALUATION

To evaluate and assess the usefulness and applicability of our business process architecture concept we designed an experiment and invited BPM experts from industry and research to participate. In the experiment the participants were asked to examine process model interdependencies and other process model information in different process model collections with the help of different business process architectures approaches. The aim was to compare our approach to other common business process architecture approaches on the one hand and on the other hand to validate if our approach fulfills the intended purpose for which it was designed. Furthermore, our intent was to gain more insights into the positioning of our approach but also other approaches in regard to the BPM lifecycle proposed by Weske [165].

EVALUATION STRATEGY: EXPERIMENT. To evaluate our approach and compare it to other approaches, we choose the research strategy of an experiment. Experiments are used to perform explanatory or causal enquiries, and compare different treatments to each other [166, 172]. The aim of an experiment is to find and explain an effect of a treatment in comparison with other treatments. Focusing on experimentation in the area of software engineering, Wohlin et al. [166] consider experiments as comparative research strategy suitable, e. g., for comparing the use of different methods to each other. Yin [172] suggests to use experiments if the focus of the study is a contemporary event, the research question focuses on the how or why, and the study requires control of behavioral events. Following Yin [172] and Wohlin et al. [166] we choose the form of an experiment as a comparative research strategy and used elements from user interface design for getting an insight of the usability of our approach, also in comparison to other approaches [143, 166]. Wohlin et al. [166] list high execution and measurement control, and ease of replication as advantages of experiments whereas the investigation costs are high.

The high control of an experiment over participants, objects, and instrumentation allows for easier generalization of the outcomes. However, there are certain threats to the validity of experiments that can be group into conclusion, internal, construct, and external validity according to Cook and Campbell in [166]. Despite that, Wohlin et al. [166] state that by following a clear experiment process including, scoping, planning, experiment operation, analysis and interpretation, and reporting these concerns can be overcome. The aim of our experiment is to measure the performance of our BPA approach and compare it to other approaches. Our experiment followed the steps of the experiment design process described by Wohlin et al. [166]. The research process consists of scoping, planning, experiment operation, analysis & interpretation, and reporting.

SCOPE OF EXPERIMENT. With the experiment we intend to evaluate if our BPA approach provides a fast and useful overview for a process model collection, allows visualizing and easily grasping complex process interdependencies, and in this regard performs better in leveraging information from process model level to the more abstract business process architecture level than common business process architectures approaches.

In terms of the goal template from Basili and Rombach cited in [166] we want to analyze our BPA approach, classification/landscape approach, and collapsed pools for the purpose of evaluation with respect to efficiency and usefulness from the perspective of the researcher in the context of BPM experts applying a PA approach.

The null hypothesis states there is no difference in performance in regard to correctly answered questions in a particular time frame among all PA approaches used in our experiment. Our alternative hypothesis for this experiment is that using our BPA approach leads to better results.

PLANNING AND EXPERIMENT DESIGN. For our experiment we choose a standard experiment form of one factor with several treatments [166]. Experiments have a clear structure and terminology. They consist of independent variables, dependent variables, objects, and subjects. Independent variables are controlled by the experiment. The effect of changing one or more independent variables is studied on dependent variables. Independent variables that are changed during the experiment are called factors. The different instances of one factor are called treatments. Treatments are applied to objects, e. g., a document or code, and subjects (participants).

We divided our experiment into two parts. The first part consisted of three BPMN process model scenarios. Each scenario represented a small BPMN process model collection with seven process models. We designed nine questions that were posed in each scenario.

The second part of the experiment consisted of seven questions on general aspects of business process architectures and BPAs especially, e. g., application areas along the BPM lifecycle as well as rankings of different approaches, and an assessment of the scenarios regarding their complexity. The second part did not follow the form of an experiment and was intended to provide some context to the experiment results and assessment of our BPA approach. For the design of the questionnaire methods for usability and user experience measuring according to Tullis and Albert [143] were used. These help to specify questions that allow evaluating the applicability and usefulness of BPAs and the other PA approaches.

We designed three simplified process model scenarios from real world examples that each represent a small process model collection. We chose real world examples to achieve more meaningful insights from the experiment. These three scenarios represent the *object* of our experiment on which treatments are applied. The *factor* of our experiment is the general PA approach and the *treatments* the concrete PA approaches used for each scenario. For comparison to our BPA approach we only used folder structures/process landscapes, and collapsed pools which appear to be the most common approaches for structuring process model repositories, but are not limited to them.

Business process architecture classification methods are often replicated in an according folder structure depicted on high level as process landscapes. The folder structures/process landscapes were used as representation of PA methods based on process model classification. Collapsed pools were used as they are part of the BPMN methodology and provide an abstraction of process models and in some sense resemble our BPA approach. PA approaches based on classification (here folder structures), process landscapes, and collapsed pools are commonly used in BPM tools [81]. In research more PA approaches are proposed as presented in Dijkman et al. [33] but could not be considered in our experiment.

The first scenario depicted the organization of a sports club. A scenario that most people can relate to easily. The second scenario was an extended version of our general use case, a construction permit application. The third scenario described the planning and execution of a logistic transport. For each scenario the participants were also provided with different business process architectures approaches. In the first scenario the participants had a process landscape as additional overview to the process models and a list of processes created with a PA classification method. In the second scenario we provided a view with all seven process models and their message flows as collapsed pools as well as the list of processes as in the first scenario. The overview for the third scenario consisted only of our BPA for the process model collection.

The scenarios were of different complexity. The first scenario was designed to be the most easy, the second slightly more difficult. The third scenario was supposed to be the most complex scenario. The reason for having an increased complexity in the scenarios was to counter bias due to the participants' learning process during the answering of the questions of earlier scenarios. This was also a reason to choose different scenarios for each PA approach.

The process models in the scenarios were structurally sound, block structured, and most of the processes in the scenarios depicted only sequences. Only few of the models contained AND or XOR blocks. The sending and receiving of messages was depicted by throwing and catching message events, as well as by sending and receiving activities. Data interdependencies or other hidden implicit interdependencies were not included in the process models. The first scenario consisted of 13 interdependencies, the second of 12 interdependencies, and the third of 27 interdependencies.

In total 20 BPM experts from research and industry participated in our experiment. These experts are the *subjects* of our experiment. Seven participants were experts from industry and covered the banking, telecommunication and insurance sector as well as a large online retailer, and a software development and consulting company. Most of them were in charge of BPM activities in their organization and responsible for the management and maintenance of the company's process model repositories. 13 participants were BPM experts from the research sector.

The process models were printed on paper so that the participants could easily flip through the models, make notes, write on them, and arrange them if necessary. The participants were given ten minutes to answer nine questions for each scenario. The questions for each scenario were the same and dealt with determining the amount of processes in a collection, independent processes, important processes, and finding interdependencies between the process models. In the following the nine questions are listed:

1. Q1: How may processes exist in the process collection?

2. Q2: How many independent (receive no trigger/messages) processes exist in the process collection?

3. Q3: Which processes are independent (receive no triggers/messages)?

4. Q4: How many processes are interdependent from more than one other process (are triggered or receive messages)?

5. Q5: Which processes are related to each other (trigger/messages)?

6. Q6: Which processes trigger which other processes?

7. Q7: To which processes do process P1 and P5 send messages (no triggers)?
8. Q8: Sort the processes in regard to their importance in regard to their number of interactions (trigger/messages) and partner processes!
9. Q9: For each PA approach assess on a scale from 1 to 5 if the following statement applies. This presentation provides the most information on the processes of the process collection?

As these nine questions repeated for each scenario the third scenario was designed to be more complex to counter the possible learning curve and avoid skewness of our experiment. We used survey monkey[1] to collect the answer for each question. We decided not to change the order of the scenarios or the provisioning of PA approach per scenario for avoiding skewedness and because of technical reasons. We were afraid to change the order of the scenarios and the given business process architectures as presenting our BPA in the beginning would have given the participants a structured way to assess the succeeding scenarios and hence skew the answers. Technically, survey monkey, the data collection tool we used, does not directly provide means for randomizing the order of the set of questions for each scenario. A manual implementation would have been an effort that could not be compensated by the expected benefit. The ninth question asking for a preference on the different PA approaches provided was introduced to measure the existence of learning effects as proposed by Tullis and Albert [143]. The *dependent variable* of our experiment is the score that a participant (subject) achieves per scenario (object) with a specific PA approach (treatment).

The time limit of ten minutes per scenario was determined by several test runs of the experiment. We stopped the time a BPM expert needed to get a good overview of the interdependencies and to answer the questions. In the course of our test runs we rephrased some questions and removed ambiguities. In a test run that we performed ourselves as experienced user, we only needed 2 minutes to construct an according BPA and needed about approximately five minutes per scenario to answer all questions, i. e., in total 7 minutes. Our initial test participants needed approximately six to seven minutes to answer the questions for one scenario with our BPA overview. Hence, we considered approximately forty to sixty percent more time for scenarios with probably less insightful PA approaches as sensible. As business process architectures should give a fast and compact overview on a process model collection, a time limit of ten minutes was considered reasonable for such a task.

1 Survey Monkey - An online survey platform for creating surveys and collecting data - www.surveymonkey.com

For the second part of the experiment we estimated a time frame of 15 min to answer all thirteen questions.

EXPERIMENT OPERATION. In the beginning of each run of the experiment the participant(s) were given a brief explanation on the structure of the experiment. We briefly introduced our BPA approach as it was not known to most of the participants. The introduction of BPA took usually only one or two minutes to explain the main concepts. If other business process architecture approaches like process landscapes or collapsed pools were not known, we also explained those methodologies in the beginning of the experiment. Then, the participants were asked to open the survey monkey link for the experiment and given the first scenario, i. e., the process models on paper plus an overview created with one of the PA approaches. At the end of the 10 min time-span for each scenario, we provided the participants with additional overviews for the current scenario created with the other PA approaches for polling their preference (question 9 of the experiment). After 10 minutes the participants were asked to proceed to the next scenario until they had answered all questions of all three scenarios. Participants were allowed to go back to previous questions only within their current scenario. Once a scenario was accomplished, the participants were not allowed to return to change answers in previous scenarios. The second part of the experiment did not have any time limits but most of the participants were able to finish it within another ten minutes. On average one experiment run took 45 min.

During the execution of the experiment we were available for questions for clarifying the understanding of the experiment or questions posed. We took notes on the questions asked and remarks of the participants for possible later usage and better reasoning about the experiment results.

ANALYSIS & INTERPRETATION, AND REPORTING. To find out if a particular business process architecture performs well in regard to giving an overview on process model collections and its inherent process interdependencies our measure was the accuracy of answers in relation to a particular time span given. The measurement was based on the amount of correctly answered questions by scenario during a fixed time span. The more correct answers given by the participant the better the performance of the according business process architecture approach. For assessments on usability and usefulness of the PA application in different stages of the BPM lifecycle we used Likert scales from 1 to 7 where the value 1 represented the worst rating and seven the best rating.

The first four questions that were asked for each scenario consisted of three single choice and one multiple choice questions. They dealt

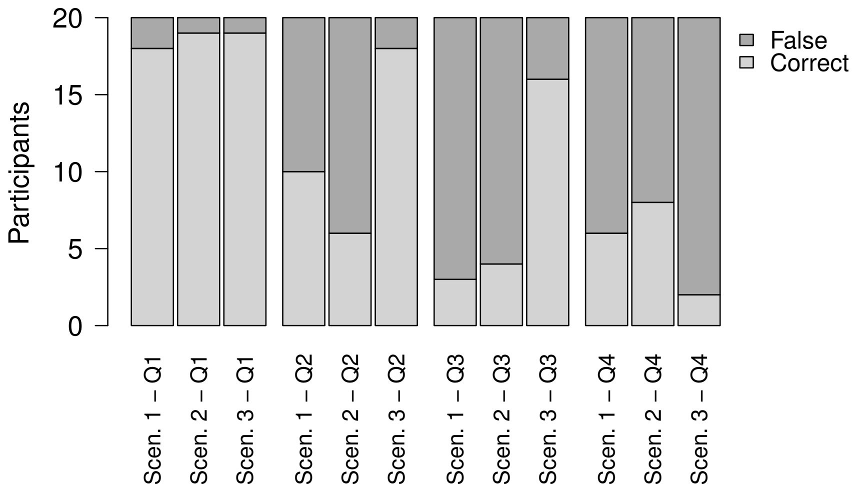


Figure 58: Results of questions 1-4 as stacked barplots

with the amount of processes in the scenario, the amount of independent process models in a given process model collection (scenario) and the identification of independent processes. These questions were meant to familiarize the participants with the process model collection at hand and provide an easy introduction into the topic before dealing with more difficult tasks.

Figure 58 shows the results represented in a stacked barplot. For each of the four questions the results for the three scenarios were grouped together. The first bar of each group shows the results for the first scenario in which the participants were given a PA classification overview and a process landscape, the second bar shows the results for the second scenario with provided collapsed pools overview and the third bar shows the results for the third scenario where participants were given only an overview created with our BPA approach.

Not surprisingly, for the first question each PA approach performs equally well with a slight decrease of the landscape approach in the first scenario.

The results of the second and third question show much better results for the use of our BPA approach where almost all participants were able to tell how many processes in the process model collection were independent and were able to identify them. For question 2 the process landscape approach surprisingly performed better than the collapsed pools.

In all three scenarios for the fourth question less than half of the participants were able to correctly answer the questions. While performing well on the first three question, our BPA approach performed worse on the fourth question where the collapsed pool approach performed best. If considering answers that missed the correct answer by one process for all three scenarios, i. e., participants iden-

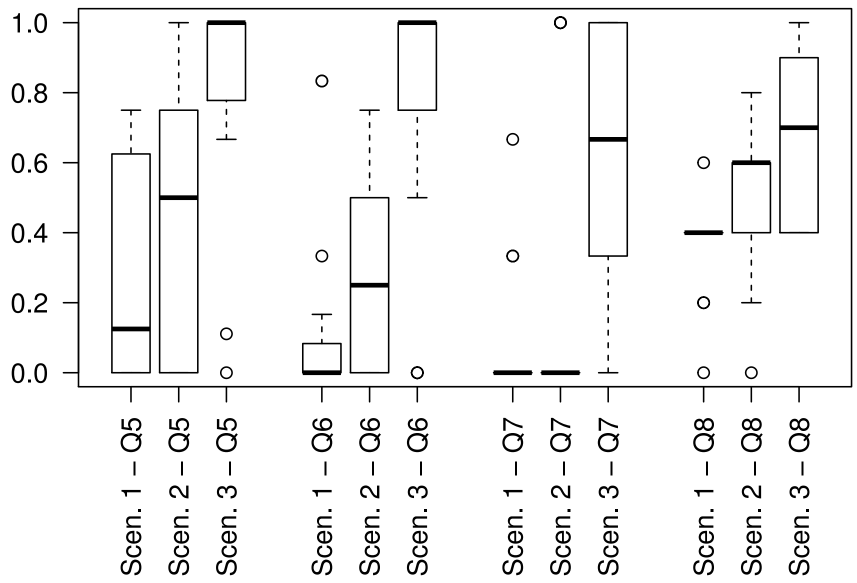


Figure 59: Boxplots for results of questions 5-8

tified one more or less independent process, the process landscape approach would perform worse and the other two approaches nearly equally well with the collapsed pool approach being slightly better. All approaches would result in half of the participants or more than half of the participants being able to correctly answer this question.

The answering of questions 5-7 required more in depth understanding of the process model collection and examine how well each PA propagates such information on higher level. The participants were given an empty 7 by 7 matrix for answering questions 5 and 6, and a 2 by 7 matrix for answering question 7 as it focuses only on the interdependencies of two processes to the other processes of the process model collection. The participants were given a point for each correct relation they identified. Question 8 asked the participants to rank the processes according to their importance considering their number of partners and interdependencies. Again, the participants were given a point for each process they put in the right order. Figure 59 summarizes the results of questions 5 to 8. We grouped the results of each question type so that the results can be compared more easily. In this regard, we also put the results into relation by mapping the scores to values between 0 and 1. The highest possible score for a question was mapped to 1 (100%) and the other scores to fractions of 1 which can be expressed in percent.

From the boxplot, it can be seen that by far the best results were achieved by the participants when using our BPA approach. The boxplots for question 5 and 6 show that fifty percent of participants reached the highest possible score with the BPA approach. The lower limit of the third quartile of the boxplots of the questions 5 and 6 of the third scenario depict that seventy-five percent were able to identify approximately eighty percent of all relations existing in the third scenario. Few outliers exist and indicate that some participants had problems with answering this question when using our BPA approach.

The PA approaches used in the first and second scenario performed worse with the increased specificity of the question posed in regard to question 5-7. The boxplot of question 5 in scenario 1, illustrates that the use of the process landscape approach resulted in the lowest values as the answer values spread widely and none of the participants reached the highest score. The results indicate that process landscapes appear not to be useful for answering these type of questions. The low results of process landscapes and collapsed pools in question 6 and 7 may be a result of lack of time as participants may have proceeded without or only partly answering these questions. None of the participants that were able to answer these questions has reached the highest possible score.

The result sets of the third scenario show that 17 or more participants were able to answer question 5 to 7 with the use of our BPA approach.

For question 8 as well, the box plots indicate that most of the participants were able to identify the correct ranking of processes or parts of the correct ranking with the help of our BPA approach. The box plot shows that about fifty percent of the participants were able to identify seventy percent or more of the correct ranking whereas for collapsed pools the maximum points where not reached and fifty percent of participants reached scores of sixty to eighty percent.

We notice that despite a specific time frame eighty percent of participants were able to answer those questions with reasonable success when using our BPA approach. Considering the complexity of the scenarios, the third scenario being assessed as the most complex by the participants in the second part of the experiment, the results support that our BPA approach meets its intended purposes and provides a good overview on the interdependencies of processes. Process landscape being used in the scenario perceived the less complex, appear not to provide useful information on the process models and their interdependencies.

The comparison of total scores of each scenario showed that also participants with low performance in the first scenario, performed well in the third scenario, similarly when comparing the overall performance of participants in the second and third scenario.

Wilcoxon signed rank tests were used to analyze the relationship between the total scores of the participants in the first scenario and second scenario with the total scores of the participants in the third scenario. We chose the Wilcoxon signed rank test as the data sets of the total scores are ordinal and do not follow normal distribution required for the equivalent parametric matched pair t-test [166]. The Wilcoxon signed rank test requires data being paired and from the same population, random and independent selection of pairs, and does not require any particular distribution of data. In both cases, the comparison of third and first, and third and second scenario, we could reject the null hypothesis in the Wilcoxon signed rank test with a confidence interval of 99%. The alternative hypothesis is true in both cases, i. e., a significant difference between the median values of the data sets was detected. The Wilcoxon signed rank test of scenarios 3 and 2 resulted in a p-value $p < 0.01$ and for scenario 3 and 1 in a p-value $p < 0.01$.

The median value of the total correct answers for scenario three is 0,82 (82%), for scenario two 0,42 (42%) and for scenario one 0,23 (23%), i. e., the performance of the participants using our BPA approach resulted in better scores. These results illustrate that our BPA approach to a large extent supports the users in reasoning on a process model collection. A possible reason for the less good performance of collapsed pools as overview, may be the high abstraction when collapsing pools, however we could not find any proof in the results.

Question 9 asked for the assessment which PA approach provides the most information on the process model collection of the scenario at hand. Besides the three PA approaches that were used in the experiment, we listed folder structures and the detailed process models at hand for evaluation in question 9 as well. To find out which PA approach provides the most information on the process model collection according to the perception of the participants and to measure the degree of consent we used the Kruskal-Wallis test that is suited for the analysis of multi-treatment experiments and also provides the mean ranks for each of the provided evaluation items. Table 6 depicts the results of this calculation. We observe that the BPA approach was ranked best among all PA approaches. The detailed process models were ranked second best as they provide a lot of detail on the process execution but provide less information on the overall picture of process interdependencies. Collapsed pools were ranked third, process landscapes fourth, and folder structures as representation of classification approaches were ranked on fifth position. The Kruskal-Wallis test rejected the null hypothesis that the result are equal with a confidence level of 95%. A significant difference between the assessments of the treatments was detected which is also depicted in the mean ranks in Table 6. The alternative hypothesis that BPA perform better was confirmed.

The results from the second part of our experiment support our findings in regard to the applicability and usefulness of our BPA approach. In the second part of the experiment the participants were asked to assess the complexity of the scenarios, rate the usefulness

Table 6: Mean ranks of assessment of PA approaches according to Kruskal-Wallis test

PA Approach	Mean Rank
Our BPA	217,49
Detailed Processs	174,33
Collapsed Pools	148,32
Process Landscapes	121,11
List/ Folder Structures	53,75

and application areas of the different PA approaches used in the scenarios.

We will summarize and highlight the most interesting results in regard to our BPA approach. Asking for the PA approach that provides the user with the most information on a process collection, our BPA approach was perceived best by eighty percent of the participants. This result affirms the trend that we could observe from the answers of question 9 where the majority of the participants perceived BPA as the approach that offers the most information on the process model collection. These results coincide with the perception of level of detail of our BPA approach by the participants. They perceived our approach having more detail than process landscapes and classification of processes in folder structures but a similar level of detail with collapsed pools.

A large part of the second part of the experiment was dominated by the following assessments areas:

- the assessment of degree of usefulness of the PA approaches in regard to particular application areas
- the assessment of well-suitedness of PA approaches for particular application area?

Figure 60 reports the results of the assessment of the degree of usefulness of each PA approach used in the experiment for restructuring processes, the identification of important processes, as starting point for analysis, and as meaningful presentation for the content of a process model collection. The x-axis depicts the values of a Likert scale from 1-5. The items in the Likert scale stand for strongly disagree (1), disagree (2), neutral (3), agree (4), and strongly agree (5). BPAs reach a score between 4.2 and 4.7 for all listed application areas where the other approaches only reach scores between 1.8 and 3.5. These results support that BPAs approach support the use cases laid out in the beginning of the thesis, i. e., *structure and manage process models*, *analyze interdependent processes*, *use of a design methodology*, and *get overview*.

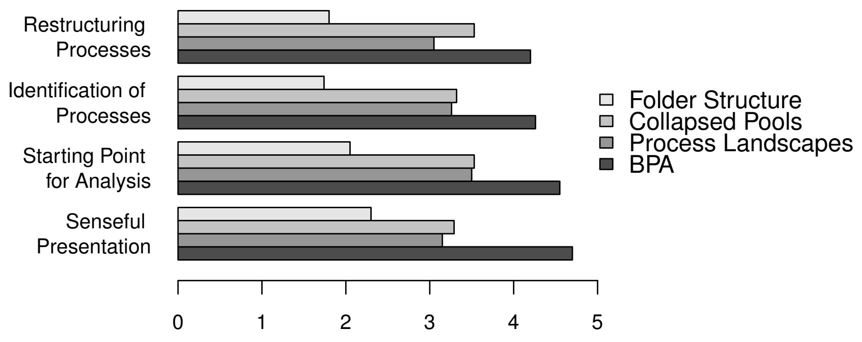


Figure 60: Bar plots showing assessment of usefulness of different PA approaches in regard to application areas

Figure 61 highlights the participants' perception of well-suited application areas for BPAs. The application and sub-domains are taken from the BPM lifecycle [165]. Each bar shows the number of participants that considered BPA well-suited for the according application area. The bar plot is divided into two parts by a larger gap, the upper part that shows the application areas that got more than fifty percent of votes of all twenty participants, and the lower part with less than fifty percent of votes.

Seventy-five up to ninety-five percent of the participants considered the presentation of the as-is-situation, analysis, the visualization of process interdependencies, providing an overview of a PMC, and documentation as top five and well-suited application areas for BPAs.

To a lower extent, but still sixty to seventy percent of participants perceived BPAs well-suited for the application areas process re-/design, process evaluation, process optimization, and process organization and process structuring. Monitoring and configuration received fifty-five to fifty percent of votes.

Except from the process evaluation and process configuration, the application areas that received fifty or more votes can be regarded as aspects of either the design and analysis phase or the enactment phase of the BPM lifecycle that we identified as our primary phases of our BPA approach in Section 1.1. The top six ranked application areas cover the BPA use cases and requirements described in Section 4.2.

The findings of degree of usefulness and findings from the assessment of well-suited application areas coincide. The top five application areas for BPAs identified by the participants can be subsumed in the four categories for which BPAs were considered very useful.

BPAs were not perceived well-suited by the participants for depicting process hierarchies, for maintenance purposes, organizational restructuring, identification of process categories, enactment, and implementation. The perception of BPAs being not suited for maintenance is surprising as BPAs should support the maintenance of business processes by showing the impact of changes on other processes.

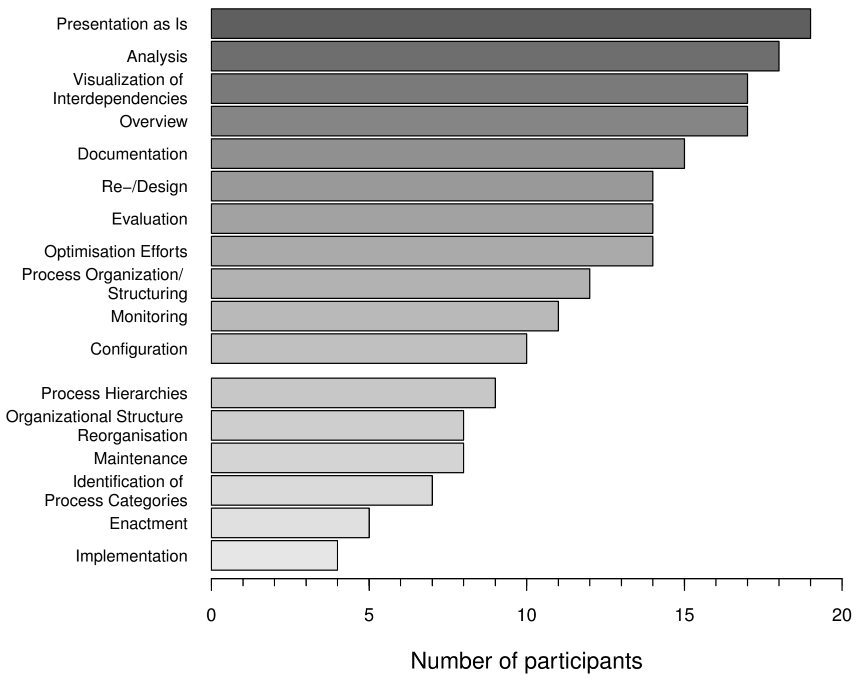


Figure 61: Assessment of BPA use in different application areas

In summary, these results show that BPAs can be used as tool providing an overview on process models, and for analyzing and visualizing their interdependencies on an abstract level. Although, only a brief introduction to BPAs was given, the participants were able to apply our BPA approach and reach good results. This indicates that our BPAs approach is easily understandable and straightforward to learn and apply.

LIMITATIONS OF EXPERIMENT. Our experiment is limited by a rather small set of participants. Despite the difficulty to find BPM experts from industry, we were able to gain a good share of BPM experts from the private sector. The length of the experiment and the need for personal assistance during the experiment limited the scale of our experiment. We intended and managed to create a heterogeneous sample of experts. Independent from the domain, knowledge experts and researchers were able to easily use BPAs and reach good results. The results obtained indicate that BPAs facilitate the understanding of complex business process interdependencies and support their analysis.

The results allowed us to find first indications on the performance of our approach and positioned them in regard to process landscapes, collapsed pools, and classification approaches. However, the results need to be treated as initial results and first indications and bear limited statistical significance. The scope of the questions covered only a limited area of PA aspects and focused mainly on the interdependency aspect of processes. In other focus areas, the other PA approaches may perform better. The size of the process model collection of each scenario was limited to seven process models but already with such a small set the difference in performance of the different PA approaches became evident. In process model collections from practice the number of interdependent processes is likely to be larger and hence the need for BPAs as well.

Despite the low sample size (number of participants), the results show a clear trend and BPAs performed well in supporting the participants in answering the questions. In all questions the participants performed better when using BPAs than when using the other provided PA approaches. The application of the Wilcoxon signed rank supported this finding and showed that there is a significant difference between the results of the BPA scenario and the other scenarios. The overall results of the experiment show that BPA fulfill the requirements we posed and realize the intended purpose they were designed for.

9.2 IMPLEMENTATION OF A BUSINESS PROCESS ARCHITECTURE TOOL

To be able to demonstrate parts of our presented BPA concepts we implemented a tool for modeling and analyzing BPAs according to their correctness criteria. The tool is part of a platform that was designed for research on process model collections. The platform serves as the basis for developing and validating algorithms on several process model collections. In the following we introduce the platform for research on process model collections. With the implementation of a modified version of our BPA extraction algorithm, we validate our extraction algorithm and the use of the platform. Then we present the architecture and implementation of our BPA analysis tool module that uses functionalities of the research platform as well.

9.2.1 *A Platform for Research on Process Model Collections*

Many approaches and algorithms towards empirical research in the field of BPM have been proposed along with several process model collections that have been made available by companies [21, 52], public bodies [42, 171], and research [71, 72]. While these collections have been subject to validate research results, it is difficult for researchers to apply their work to different collections—a main obstacle that hinders empirical evaluation. This is due to the heterogeneity of the structure of these collections, their format, and modeling language, which yield a considerable overhead to explore, transform, and extract relevant information.

To reduce work and foster empirical research in the context of process model collections, we developed an analysis platform that aims at allowing researchers to focus on their actual research questions, rather than spending time and effort pre-processing data from heterogeneous sources.

The platform provides uniform access to a large variety of process model collections, by means of import functionality that allows capturing any kind of information related to a process model. Utilities facilitate iterating over process models, filtering them, segmentation and extraction of information, as well as transformation into a generic representation, jBPT[2], that covers common concepts of process modeling languages. In our case, the platform allows us to apply our BPA concept to several process model collections. As the platform is an open source project[3], maintained by a scientific community, it is accessible to everyone and open for extensions.

The platform is not a process repository to actively manage process model collections, but an analysis platform, which provides the fundamental infrastructure (including a uniform access to different process model collections) for analysis. Since the development of the platform in 2012 the amount of analysis modules has grown due to contributions of students and other researchers. A substantial amount of analysis modules is provided, for instance, process metrics calculation [97] as well as process collection clustering capabilities. With the increase of shared analysis modules and process collection importers, implementation work for the individual researcher dealing with a particular interest decreases as existing modules can easily be reused. In the development of our BPA tool we utilized some of the existing platform utilities. Building on our BPA tool, Breske [15] integrated his BPA extraction algorithm into the platform as well. We elaborate on the requirements and architecture of the platform although it focuses not directly on BPAs but provides fundamental functionality and data to test and validate our BPA approach.

9.2.1.1 *Requirements for the Process Model Collection Analysis Platform*

To provide a platform for research on process model collections bears several challenges and particular requirements to the software architecture of the system. The requirements derive from the various process collections that differ in structure, process model representation, and content on the one hand, as well as current research questions on the other hand. The main requirements for a platform are the management of heterogeneous process model collections, the filtering, segmentation, storage, and modular analysis, as well as the support for reuse and repeatability.

2 http://code.google.com/p/jbpt/
3 http://bpmai.org/BPMAcademicInitiative/BpmTools

Management of heterogeneous process model collections. To date, there are few process model collection available for research. One of the most commonly used collections is the SAP reference model [21], published in 1999, that comprises 604 EPC, process diagrams that describe the SAP R/3 system. Some time ago, IBM released[4] a collection of 735 business process models from the Websphere process modeler available as BPMN [112] process models, which cover various industrial domains, e. g., finance and telecommunications [52]. In the course of the BPM academic initiative[5] (BPM AI), a set of *22651* process models in the time of writing (2014) created by students in various process modeling languages has been made available to the research community [71].

These process collections have been created with different intentions depending on the organizational domain and on the modelers' level of BPM expertise. As a consequence, process collections differ, among others, in process modeling languages, e. g., BPMN, EPC, and Petri nets, file format, and accompanying metadata. For instance, all models from the SAP reference model are stored in one large EPML file; each model from the BPM AI, in contrast, is represented by a JSON file that captures the model graph and an SVG file that contains a visual representation of the model. In the national process library [42] many process models have no graphical representation at all, but they are classified by a set of structured metadata and described in prose. As one process model can appear in different forms, e.g., as in the case of BPM AI with JSON and SVG representations, the platform shall be able to store several representations of one process model and always keep the original sources.

Versioning engineered artifacts is good practice in order to track changes and revert mistakes; this also holds true for process models and is supported by several BPM tools. Hence, it is compulsory for our platform to preserve version information, too.

Filtering, segmentation, storage, and modular analysis. Above characterization of the heterogeneity of process model collections requires a modular approach to discover relevant aspects of process models in a large collection for analysis. Therefore, it is desirable to filter the set of process models of one or several process collections, transform them into a uniform representation, and extract particular features (segmentation) for consideration. This allows, for instance, obtaining only event and activity labels for discovering process model interdependencies and later extraction of BPAs.

Analysis of process models and collections should be modular in order to facilitate reuse of certain functionality across different experiments and allow researchers to benefit from each other more ef-

4 http://www.zurich.ibm.com/csc/bit/downloads.html
5 http://bpmai.org

fectively. By this, more complex analysis can be constructed by assembling several atomic analysis modules and execute them over a filtered and segmented set of process model data.

Researchers, who work on large data sets and computation intensive tasks, may run experiments over several hours or days. The platform needs capabilities to store analysis results relating them to particular process models and providing means to efficiently obtain this data again.

SUPPORT REUSE AND REPEATABILITY. Research typically builds on former research results, refers to it, extends it, and improves it. In a similar way, the platform shall allow easy (re-)use of analysis functionality and help researchers to cooperate with each other. Modular analysis modules are one step into this direction.

Methods or innovations derived from analyzing one process model collection are only valid for the context of that collection, i. e., they may not be valid in other cases. Hence, validation beyond a single collection requires application of the same procedures to other collections. A uniform representation of graph-based process models provides a solid basis for analysis. This enhances the platform to use one format to iterate over different process languages on the one hand. On the other hand, analysis modules can be re-used in an easy way and do not need to be re-implemented for different process representations. Thus, a structure is required that is flexible enough to map different process modeling notations but not too generic to reduce them to meaningless blocks.

9.2.1.2 *Platform Architecture*

According to above requirements, we identify three main functional areas in the conceptual architecture of our platform, illustrated in Figure 62: Import, analysis, and index management. We briefly describe each feature in the context of the architecture.

The *import* module extracts distinct process models from the original collection and imports them into the platform, mapping them to a flexible data schema to capture all information provided. *Filter management* contains compact units of functionality to iterate over imported process model collections, filter, segment, and transform process models and supply the extracted information into the *analysis* modules provided by the respective researcher. Finally, *index management* allows storing analysis results related to process models in a key-value store and obtaining stored data by means of *querying* that data.

In addition to the core functionality, the platform relies on a *process model repository* to store the process models and analysis results. The actual repository interface is wrapped by the *persistence API* that maps interfaces offered by the repository to interfaces required by

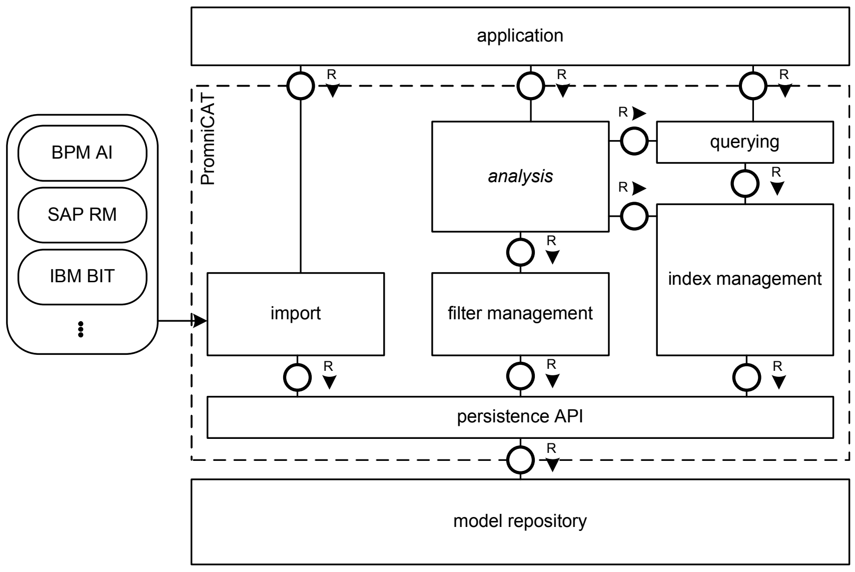


Figure 62: System architecture of the platform

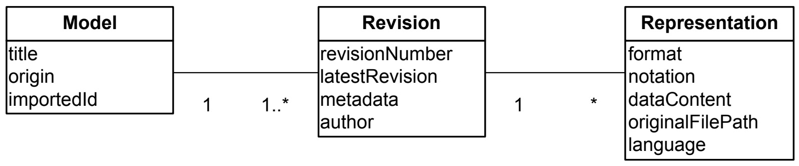


Figure 63: Process model data schema

the platform's modules. In the present case, we resorted to a graph-document database management system, that is, OrientDB[6].

9.2.1.3 *Data Schema and Import*

The data schema provided by the persistence API is presented in Figure 63. Here, a *model* stores all information that is related with one logical process model from the collection. Each model can have several *revisions*, based on our observation that each process model can be changed over time and that these changes may be relevant to research. Each revision, in turn, can have any number of *representations*. If models are not version-controlled in the collection, a model will be linked to exactly one revision in our data model. Each revision is identified by a *revision number*, its *author*, and a flag (*latestRevision*) that identifies the most recent revision.

For each process model, its *title*, *origin*, and an identifier (*importedId*) are stored. The origin indicates the process model collection from where a model has been imported. The *importedId* is used to correlate imported models with models from the collection.

6 http:\www.orienttechnologies.com

As introduced in Section 9.2.1.1, we do not restrict the type of data to be analyzed. Hence, our platform allows importing almost any kind of process descriptions independently from their format, structure, properties, and the capability to be mapped to a generic process representation. This is achieved by relating a process model with one or many representations, i. e., process descriptions in their original format, see Figure 63. Since we do not restrict the data to be stored, the original process representation imported from the collection is stored as byte array in our model repository in *dataContent*.

As various process collections are structured differently, a separate import module is required for every kind of process collection to be imported. Currently, the platform provides importers for the different process collections presented in Section 9.2.1.1, i. e., an EPML importer (for the SAP reference model), an importer for BPMN models in the standard XML representation (for IBM BIT models), and an importer for models from the Signavio process modeler that has been used to create models of the BPM AI.

Importers also support update functionality to synchronize an already imported process model collection with a more recent version of this collection. An update would then only add new models, new revisions, and additional representations, rather than changing stored representations to avoid invalidating existing analysis results. However, it is also possible to import a previously imported collection again by giving it a new *origin*.

9.2.1.4 *Model Analysis*

The central purpose of our platform is to provide a basis for model analysis, realized by the interaction of *filter management* and *analysis*, illustrated in Figure 62. Technically, the analysis module is rather a collection of several units that offer particular algorithms to examine models, sets of models, or aspects thereof. These analysis units are to be developed by researchers with regard to their research question and requirements. Nevertheless, we already provide some analysis units, e. g., a subset of the metrics that measure process complexity presented in [97]. To release researchers from the burden to choose models from a collection, extract relevant information, and transform it into a format applicable for the respective analysis units, the platform provides so-called filter chains, following the pipes and filter integration pattern [61]. Here, filters serve as utilities to load and handle process models from the repository and can be categorized by their function to filter, transform, or extract information.

A schematic example for filter chains is given in Figure 64. Access to the repository is always provided by a *DatabaseFilter* that allows setting filter criteria on the data model presented in Figure 63. For each model, the database filter provides its database id and the loaded model representation. Attached to the *DatabaseFilter* follow *Filter Units*

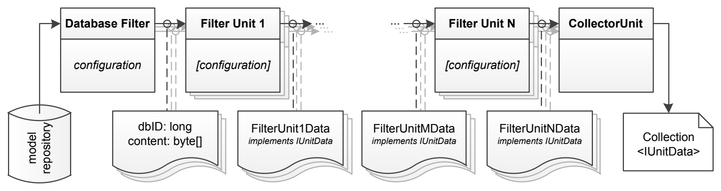


Figure 64: Example filter chain that extracts labels from a BPMN process models

that accept the output data from the previous filter as their input, select, transform, or extract content, and provide their result to the subsequent filter. Input and output data of a filter represent information extracted from one model, and must implement the interface *IUnitData*. Only filters that match in their input and output data type can be connected. For each model, this chain is separately executed, and several chains can be executed in parallel to increase processing performance on multi-CPU computers. Finally, the *collector* consolidates the results of every chain and provides them to the analysis module.

The platform already provides a set of filter units readily usable by researchers, for instance, the aforementioned database filter. Further units provide functionality to segment process models into particular concepts, e. g., extract all activities of a process model.

As we encourage reuse of implemented functionality within the platform, we envision analysis modules to be implemented as generic as possible. By that, analysis can easily be applied to different process model collections to repeat previous experiments and compare results. That is, whenever possible, analysis modules should be built on a generic process representation.

Therefore, our platform provides a filter unit that parses process models to a representation in the *jBPT* format. jBPT is a Java-library[7] that leverages graph structures to support a canonical process representation providing descendants for each supported modeling language. Besides, this library offers a comprehensive set of techniques to transform, verify, and analyze process models, e. g., provide transformation of process models to Petri nets, soundness checking, net unfoldings, or workflow graph decomposition into process structure trees. Algorithms that work on the canonical representation can also be applied to descendants. For every type of representation that shall be imported, a corresponding parser is required that transforms the model into the jBPT graph format. Currently, transformations for EPC and BPMN process models from the SAP reference model, IBM BIT, and BPM AI model collections are available.

7 http://code.google.com/p/jbpt/

9.2.1.5 *Indexes.*

Analysis results need to be stored within the platform by simple means. On the one hand, this frees researchers from the need to maintain their own persistence mechanism to store results; on the other hand, it allows referring to stored process models while maintaining consistency of linked models.

As the platform addresses analysis of process model collections, we envision analysis results also to be collections of data, likely categorized along certain dimensions. Therefore, we opted for a key-value store that allows relating any kind of Java data to a key that is either a string or a number. Due to the character of this *m:n* mapping, where one key may relate several data entities and one data entity may be identified via several keys, we refer to the storage as an index. The querying module provides means to select a subset of stored data by querying over keys.

9.2.1.6 *Platform Extension BPA Extractor Module*

We validate our extraction algorithm and showcase the use of the process model collection research platform with the implementation of a BPA extraction module. The BPA extraction module was implemented as part of our research platform by Breske, a master student under our supervision [15], whose work was briefly sketched in Section 7.2.6. The BPA extraction module implements a modified version of our BPA extraction algorithm described in Section 7.2.

For this, Breske integrated the BPA concept into the jBPT class hierarchy. To adapt our BPA concept of many interacting models to the jBPT process model description a BPA model is considered a jBPT process model, and a BPA process is considered a NonFlowNode. The different BPA event types are considered FlowNodes. The concept differentiates between *InterFlow* and *ExternalFlow* for being able to map information flow and trigger as well as the sequential internal control flow of BPA processes. The mapping of our BPA concept to the jBPT process model facilitates the extraction and transformation of process models to BPA processes. The mapping allows using the research platform's utility units and applying different jBPT analysis functionality to BPAs. Being provided with parsers and process model filter units, the BPA Extractor module concentrates on the implementation of the BPA extraction algorithm.

The BPA extraction module consists of the main classes *BpaExtractor*, *BpaDependencyAnalyzer*, *RpstNodeAnalyzer*, and *ModelInterdependencyAbstractor*. The *BpaExtractor* is the main module responsible for the extraction of BPAs from a set of process models [15]. It executes the extraction algorithm by performing the according three steps; process model abstraction, creation of BPA elements, and interdependency association. The process model abstraction step removes insignificant elements, i. e., nodes that do not take part in a process interdependency. After analyzing the process models with the help of the according RPST_{PM}, the algorithm creates BPA processes and events. It then connects the events of the created BPA processes by inserting triggers or information flows according to the interdependencies found. The BpaExtractor uses the classes *ChainBuilder*, *RPST*, *RpstNodeStructure*, *RpstNodeType*, and the interface *IUnit* provided by the research platform and jBPT.

9.2.1.7 *Showcase: The BPA Extractor Unit Chain*

We showcase the use of the research platform's utility units with the newly developed analysis module, i. e., the realization of the BPA extractor module. Utilizing this approach, we focus on the support the platform provides to access and extract desired elements from process collections rather than the BPA extraction algorithm itself.

The algorithm consists of three steps, interdependency detection, process model abstraction, and the process model extraction. Interdependencies between processes are determined by label matching. Hence, all nodes that do not have a matching label with a partner node of another process are removed from the process models. The abstracted models can then be analyzed, their RPST_{PM}s created, and the according BPA process generated.

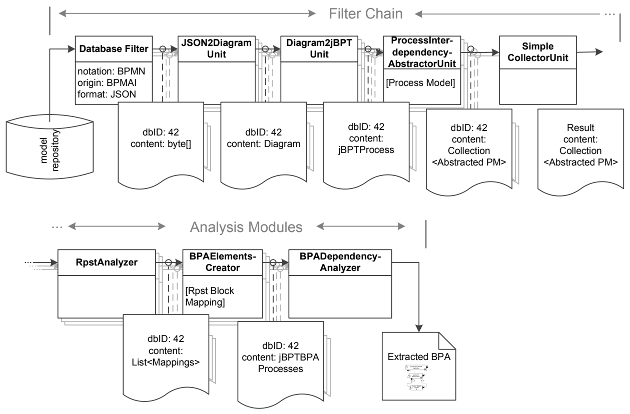


Figure 65: Showcase unit chain and analysis modules

To perform the extraction of BPAs from process models we need to set up a utility chain that provides us with the abstracted models. This is done with the *CorrectionUnitChainBuilder*. The *CorrectionUnitChainBuilder* repairs process models that may not correctly be modeled or have small flaws and assures the quality of the process models that are fed to succeeding utility and analysis units. Figure 65 shows the utility chain.

All utility units implement the *IUnitData* interface. Not all of the filter utility units can be put in direct sequence, because each consumes specific input and produces specific output data.

First, we need to create and configure the database filter. This is done by creating a new *DbFilterConfig* and adding origin, format, and notation to it. As we are interested in BPMN process models from the BPM AI collection, we set *BPMAI* as origin value, *BPMAI_JSON* as format value, and *BPMN* as notation value accordingly.

Then, the database filter is added to the *CorrectionUnitChainBuilder*. In the next step, the process models must be mapped into the generic jBPT format to enable the detection of nodes of significance and the removing of insignificant nodes. Therefore, we register the *BpmaiJson-ToDiagramUnit* followed by the *DiagramToJbptUnit*. The sequence of those two units performs the transformation from the BPMAI-JSON process model representation to the jBPT representation of the process model.

So far we only utilized existing functionality provided by the platform. The *ProcessInterdependencyAbstractorUnit* was newly developed to detect the nodes that interact with nodes of other processes and remove all other from the process models. It handles data objects and events. Gateways are kept for reasoning on the control flow that impacts the later aggregation of nodes, the generation of BPA events, and the mapping between the aggregated nodes and BPA events. The output of this unit are abstracted process models.

Finally, a *SimpleCollectorUnit*, provided by the platform, is added to the unit chain to collect and consolidate the result of the utilized unit chain. In each step of the unit chain, the reference to the original process model is preserved. Listing 4 shows the code required to set up above filter chain and demonstrates that only little effort is required to access the desired aspects of process model collections for analysis.

Listing 4: Code excerpt for filter chain for extracting BPAs

```
Collection<UnitData<Object>> abstractedModels;
CorrectionUnitChainBuilder chainBuilder = new
    CorrectionUnitChainBuilder("configuration.
    properties", Constants.DATABASE_TYPES.
    ORIENT_DB, UnitDataJbpt.class);

// build db filter
DbFilterConfig dbFilter = new DbFilterConfig();
dbFilter.addOrigin(Constants.ORIGINS.BPMAI);
dbFilter.addFormat(Constants.FORMATS.BPMAI_JSON);
dbFilter.setLatestRevisionsOnly(true);
dbFilter.addNotation(Constants.NOTATION_BPMN2_0);

chainBuilder.addDbFilterConfig(dbFilter);
// transform to jbpt units
```

```
            chainBuilder.register(new BpmaiJsonToDiagramUnit
                ());
            chainBuilder.register(new DiagramToJbptUnit(TRUE)
                );

            // abstract model unit
            chainBuilder.register(new
                ProcessInterdependencyAbstractorUnit());

            // result collector unit
            chainBuilder.createSimpleCollectorUnit();

            // run chain
      abstractedModels = (Collection<UnitData<Object>>)
          chainBuilder.getChain().execute();
```

Having extracted and processed the process models necessary for the extraction of BPAs, the models are analyzed and the remaining nodes in the process model aggregated and mapped to BPA processes, their events, and their interdependencies. To achieve this, $\mathrm{RPST}_{\mathrm{PM}}$ are generated and analyzed by the $\mathrm{RPST}_{\mathrm{PM}}$*Analyzer*, depicted in Figure 65. The $\mathrm{RPST}_{\mathrm{PM}}$ blocks are used to map their nodes to specified behavior classes. Using this information the *BPAElementsCtreator* module creates the BPA processes with the according amount and sequence of events specified by the behavior classes. In the last step the *BPADependencyAnalzer* maps the found interdependencies to trigger and information flow on BPA level based on the abstracted $\mathrm{RPST}_{\mathrm{PM}}$ blocks to BPA event mapping.

9.2.2 *Business Process Architecture Analysis Module*

Modeling guidelines were introduced to improve and harmonize the quality of process models created by different process modelers in an organisation [97, 101]. For single processes several tool-supported approaches exist which allow to check structural, behavioral, and linguistic properties. Lately, these were also incorporated into modeling tools, e. g., the Signavio BPM tool[8]. Similar approaches, taking a holistic view for assuring quality on a higher abstraction level do not yet exist. Business Process Architectures (BPA) and their correctness criteria present a novel approach to organize business processes in a PMC and analyze them as introduced in Chapter 6. To model BPAs and to decide the correctness of process model interdependencies is not supported by business process modeling tools.

In the following, we present a novel and innovative tool to visually model BPAs and analyze them for correctness. The tool consists of a BPA core module that integrates existing applications that it builds on and extends it for our purpose of BPA analysis. The analysis of BPAs

8 http://signavio.com

provides a first step for assuring further correctness and consistency properties on a more detailed process layer.

Our BPA tool extends and composes functionality of existing tools as depicted in Figure 66. The user interface for modeling BPAs and visualizing found errors is provided by an extension to the Signavio Core Components (SCC)[9]. They are the open source components of the Signavio editor, a web based business process modeling tool widely used for teaching in academia[10] (BPM Academic Initiative) and as commercial BPM tool[11]. The Signavio Core Components were se-

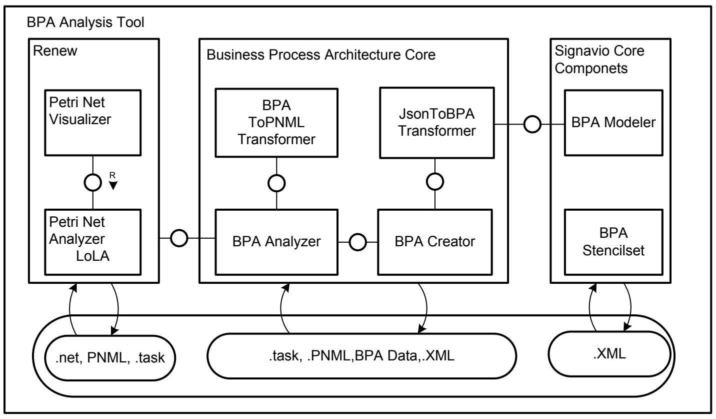


Figure 66: BPA tool architecture

lected because they provide an open source web-based editor, that is easy to extend and allows to export BPA diagrams as XML files. The Signavio Core Components can be easily extended with new modeling notations by defining stencil sets. A stencil set describes the syntactical structure of a modeling notation by defining the elements of the notation, and the containment and connection relations between the elements. Our extension introduced a new stencil set for BPAs, which contains visual shapes and connections rules to draw BPA diagrams. The editor provides a plugin mechanism to add extra functionality for a modeling notation. The BPA analysis functionality is called through such a plugin.

The main program logic is implemented as a module for our PMC research platform[12], presented in Section 9.2. This BPA module consists of a BPA data model, the BPA Analyzer, and two transformation modules, the *JSONToBPA Transformer* and *BPAToNet Transformer*. The data model defines the structure of a BPA, its processes, events, and the trigger and flow relations between the events. The *JSONToBPA Transformer* imports the .XML file output by Signavio Core Components, extracts the JSON code, and creates the BPA data structure as modeled in the diagram which is then provided to the *BPA Ana-*

9 http://code.google.com/p/signavio-core-components
10 http://bpmai.org
11 http://signavio.com
12 http://code.google.com/p/promnicat

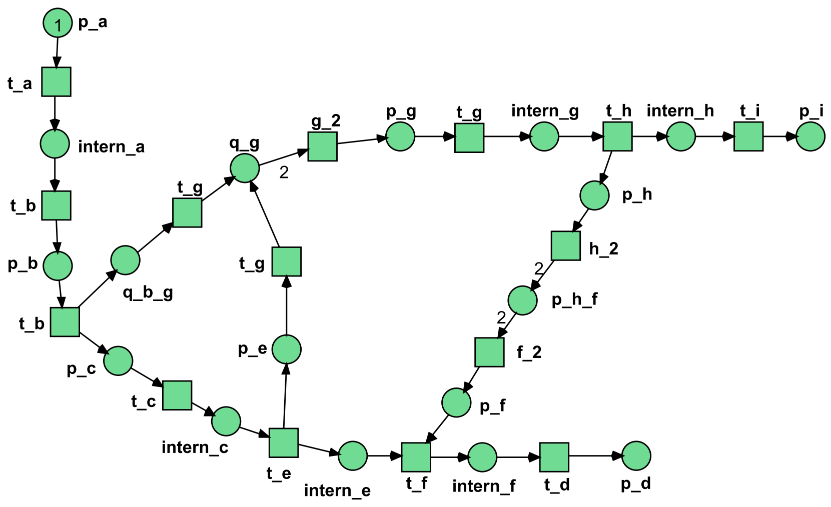


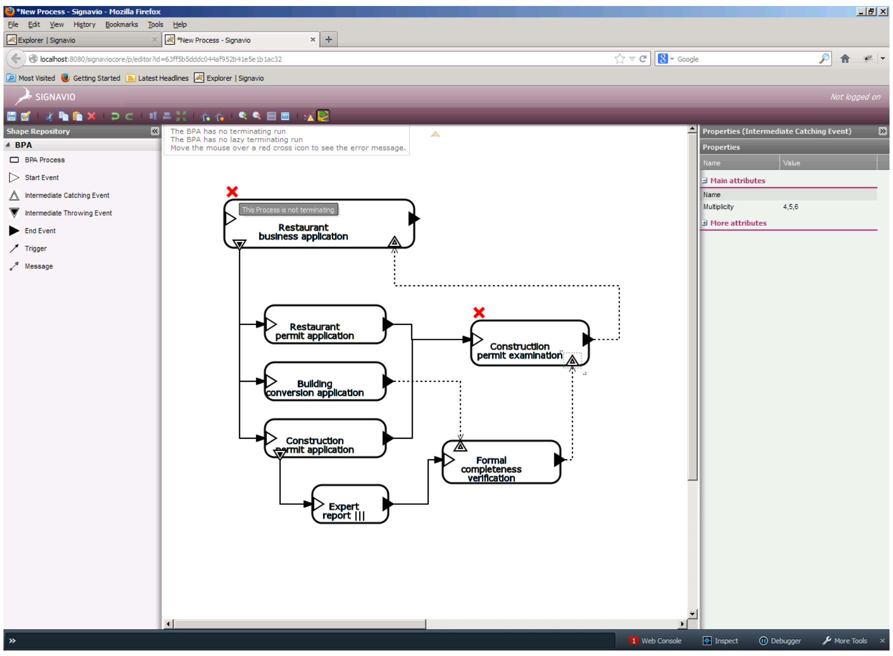
Figure 67: Open-net visualizer

lyzer. The BPA Analyzer requests the transformation of the BPA data structure into .net format from the *BPAToNet Transformer*. Taking the data structure as input the *BPAToNet Transformer* transforms the BPA into a set of open nets according to the approach presented in Section 5.2. Afterwards, the nets are composed and the resulting net is serialized using the .net standard that LoLA[13] requires as input. The *BPAToNet Transformer* also generates a set of CTL formulae which express the correctness criteria for the given BPA and are to be checked by LoLA. The transformation of the CTL formulae is performed during the transformation and composition of the open nets. The CTL formulae are derived by examining the pre- and postsets of all events in the BPA, e. g., if an end event has an empty postset, the event is considered part of the final marking of the overall net. According to the CTL formulae LoLA will check for different correctness properties, e. g., if a given BPA deadlocks, is terminating or lazy terminating.

In addition we use the Petri net simulator RENEW[14]. The Renew module provides a built-in LoLA integration for the analysis and allows for the visualization of the transformed open nets. The *BPA Analyzer* calls LoLA with the open net and the CTL formulae specified for the correctness analysis. An example of the visualized open net is shown in Figure 67.

The result of the model checker is finally interpreted and visualized in the Signavio web editor module. If all formulae yield a positive result, we know, that the BPA is correct and the according message is displayed in the upper left part of the screen. Otherwise, errors found are visually displayed as red cross on the BPA diagram in the web interface. A text explaining the errors found can be displayed by hovering over the visualized errors. The errors found are visualized

13 Low Level Petri net Analyzer, `http://service-technology.org/lola`
14 `renew.de`


*New Process - Signavio - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Explorer | Signavio
*New Process - Signavio
localhost:8080/signaviocore/p/editor?id=63ff5b5dddc044af952b41e5e1b1ac32
Google
Most Visited Getting Started Latest Headlines Explorer | Signavio
SIGNAVIO
Not logged on
Shape Repository
BPA
BPA Process
Start Event
Intermediate Catching Event
Intermediate Throwing Event
End Event
Trigger
Message
This Process is not terminating
Restaurant business application
Restaurant permit application
Building conversion application
Construction permit application
Construction permit examination
Formal completeness verification
Expert report
Web Console
Inspect
Debugger
More Tools

The BPA has no terminating run
The BPA has no lazy terminating run
Move the mouse over a red cross icon to see the error message.

Properties (Intermediate Catching Event)

Properties

Name	Value
Main attributes	
Name	
Multiplicity	4,5,6
More attributes	

Figure 68: BPA analysis tool screenshot

with a red X on the screen. A summary of the analysis and errors is also shown on the upper left part of the screen. Figure 68 shows a screenshot of the implementation and the summary of the BPA analysis. We modeled and analyzed the use case on enterprise founding application with construction permit from Section 6.3.

10 CONCLUSIONS

This chapter provides a summary of the main contributions of our thesis and concludes this work. First we will summarize the results, i. e., a short description of the formal BPA concept, its structural and behavioral properties, analysis techniques, and our overall BPA methodology including its BPA extraction and decomposition algorithms. We discuss each contribution and present limitations of our work and techniques developed. In this regard, we also highlight opportunities for future work on BPAs.

Business Process Management (BPM) has become a commodity for organizing and improving an organization's operations in the public and private sector increasing the efficiency of their business processes. In the course of BPM projects in companies and organizations large process model collections have emerged and with their existence new challenges in regard to their organization and structuring. Business process architectures are a tool to organize and structure business process models within a process model collection and define their relationships with each other.

Common PA approaches have mainly focused on the classification of business process models along functional, action-based, goal-oriented, business object-based, and similar aspects. This resulted in hierarchical classification visualized in folder directories or other navigation layers. Hence, the interdependencies found in those PA approaches were mainly of hierarchical nature, e. g., *generalization*, *specialization*, or *consists of* relations.

Only very few PA approaches allow specifying horizontal relations between processes although those are necessary for providing a holistic overview on the business processes and their influence on each other, e. g., for adjusting their interaction and interfaces. The larger context depicted by all interdependent process models is important for reasoning on the impact of changes, identifying important processes, and estimating resource requirements as well as evaluation process cost and other aspects of interest.

In this thesis we introduced a novel PA approach that specifies and visualizes horizontal interdependencies between business processes. This is the basis for reasoning about a large group of interdependent business processes. Our approach complements existing PA research and supports the design and analysis as well as to some extent the enactment phase of the BPM lifecycle.

In the remainder of this chapter we will summarize the main contributions in Section 10.1 and discuss them in Section 10.2. In Section 10.3 we introduce areas of future work based on the limitations presented.

10.1 SUMMARY OF RESULTS

We observed gaps in PA research in Chapter 3 that concern specifying and visualizing horizontal process interdependencies, missing formalisms and analysis techniques, and a general narrow focus on the classification of single process models. In this regard, we contributed a PA approach that is formal and allows specifying, visualizing, analyzing horizontal relations (trigger, message, data) between processes. Our BPA approach encompasses extraction and decomposition algorithms and can be applied top-down and bottom-up. For the extraction and decomposition we designed consistency criteria between process model level and PA level which many of the current PA approaches lack. Those aspects form the main contributions of our research:

- A formal foundation of our BPA approach, i. e., a clear specification of the elements, their relationships, and semantics (Chapter 4, Chapter 5).
- The definition of structural and behavioral correctness criteria for our Business Process Architecture approach (Chapter 4).
- Unique Business Process Architecture analysis techniques that allow analyzing BPAs for their correctness and enable quality improvement of business process model collections (Chapter 6).
- An overall top-down and bottom-up BPA methodology including BPA extraction and process model decomposition algorithms (Chapter 7).
- A Business Process Data Architecture extraction algorithm considering data objects for identifying process interdependencies between business processes (Chapter 8).

BASIC BPA CONCEPT. In contrast to many other PA approaches that focus on the classification of single processes our PA approach addresses horizontal relations and expands the scope to multiple processes putting each process model in the context to its interdependent processes. This allows assessing, examining, and optimizing the impact of the re-design of business processes in such a network, and visualizing and reasoning about inherent process model metadata like required resources or costs along many processes.

Our BPA concept defines the elements of a PA, business processes, their interdependencies, and their semantics. A business process of our BPA is a set of events. Process interdependencies are reduced to two relation types, trigger and information flow relations, that are defined between the events of different processes. Trigger and information flow relations portray the triggering of processes, respectively, the message exchange during process execution. Those encompass horizontal relational concepts found in other PA approaches as "uses" or "activates" [86].

We chose events as they express the exchange of triggers or information flows in many business process modeling notations. Generally, any other element that expresses an interdependency with another process can be mapped to events. Events are placeholders for elements that express interdependencies, e. g., sending activities in Section 7.2.1 or data dependencies introduced in Chapter 8.

Our BPA approach provides a simple and basic but formal tool to connect business processes and model their complex process interdependencies, such as multi-communication, on an abstract level that can easily be applied and understood by practitioners. The formal description is the foundation on which the other contributions of this thesis build.

definition of structural and behavioral BPA properties. The formal foundation of our BPA approach allows specifying concrete structural and behavioral properties. Other approaches do not specify any properties in regard to their structuring and organization system. None of the reviewed PA approaches state concrete quality characteristics for designed PAs. The surveys [84, 85, 35, 54, 56, 53] name required properties and characteristics for the evaluation of PA approaches but they do not specify clear characteristics for the actual implemented PA of a process model collection. We introduced the notion of well-formed BPAs, a structural property, and a notion of BPA correctness that encompasses behavioral BPA properties like BPA termination, deadlock freedom, or freedom of dead process. On PA level, our BPA approach is the first to specify desired properties for a complex system of interdependent processes that represent an organization or a company.

On process model level, we find similar behavioral notions like local enforceability, realizability, weak soundness or global termination for choreography approaches [27, 23, 90]. Although serving as example, those choreography approaches focus on the inter-organizational interaction between actors and service composition and refinement. They do not directly allow modeling mult-communication, broadcasting, or multi-instance triggers. The structural restrictions on the involved process models and on the sequence of interaction, e. g., local enforceability, are too strict for the realm of BPAs where business processes within a company can be loosely coupled and act more independently. In contrast to choreographies a strict public contract between two cooperating entities is not necessary as the processes belong to the same organization. Based on the definition of BPA properties the quality of a BPA can be assessed.

BPA ANALYSIS. For improving the quality of a process model collection and harmonizing interdependent business processes, we introduced two analysis techniques. The first technique provides a range of regular patterns and anti-patterns presenting desired and undesired interdependencies in a BPA. This approach is especially useful for the analysis of process interdependencies in synchronous environments. This analysis technique is limited by its scope to direct interdependencies between two processes at a time.

The second analysis techniques is based on the well-known Open net (ON) formalism. We introduced a transformation from BPAs to ON. For this, each BPA process, and all trigger and information flow relations are transformed into ONs. At the same time the correctness criteria for the according BPA are transformed into CTL formulae and state predicates. Both the ON and the CTL formulae are then used to verify the BPA with current model checking tools like LoLA.

The analysis of BPAs allows finding problems early in the design phase caused by insufficiently adjusted process interdependencies. In this way, time and costs can be saved by adjusting the process interaction on high level before having to re-model a range of business processes on the process model level.

BPA METHODOLOGY. The BPA methodology describes the concrete application of our BPA approach on a process model collection top-down, e. g., at the beginning of a BPM project, or bottom-up, e. g., when a process model collection already exists. To provide a clear link between process model level and BPA level we introduced consistency criteria between the BPA and business process model level.

The top-down approach includes a decomposition algorithm that creates process model skeletons from the BPA model. The skeletons can be further refined until the desired granularity is met. The refinement must comply with the BPA model on top level in regard to its interdependencies. This is reflected in provided consistency criteria.

The top-down approach envisions to model and analyze the interdependencies between business processes as early as possible to avoid costly re-adjustment at a later stage. The process models on process model level are only generated and refined after the modeled BPA is correct.

The bottom-up approach relies on an existing process model collection. It defines an BPA extraction algorithm that parses the business process models, identifies their interdependencies and multiplicities, and generates a BPA for the process model collection. In this way an overview that visualizes the interdependencies is created automatically. The extraction algorithm supports the maintenance of the BPA as every change made on process model level can be propagated onto BPA level and the BPA is automatically updated. Changes could be visualized by highlighting new elements or removed elements as often found in process modeling tools, for example.

10.2 LIMITATIONS AND DISCUSSION

BPA FOUNDATIONS AND PROPERTIES. The BPA specification is reduced to few basic elements and relationships. We concentrate on the horizontal relations trigger and information flow. The hierarchical relations between the process models on process model level and the according business process representation on BPA level are not explicitly stated in the formal BPA definition in Definition 29. They are described in the basic and complex extraction algorithms, which relate one process model on process model level to one business process on BPA level. The BPA process is a generalization of the business process represented on the process model level.

In contrast to the common single process model classification approaches, our BPA approach considers multiple processes that groups them according to their interdependencies in BPA subsets that depict a concrete scenario. The application of our BPA approach to a process model collection without any interdependent business processes would result in a set of minimal BPA subsets that each contain only one process model. In cases that a process model collection does not contain any or only few process interdependencies, the current hierarchical classification approaches yield a better structure for a process model collection. Our BPA approach is complementary to current hierarchical PA approaches and extends them with a horizontal view across their hierarchical structures. Each PA approach provides a different entry point into a business process model collection and view on its valuable information.

We hide the complex inner logic of business processes and straightjacket it into sequential BPA processes to create a simple overview on the complicated interdependencies between business processes. This simplification creates implications on the process models and the extraction and decomposition algorithms. In order to leverage the influence and strengths of the interdependencies, we limited our approach and algorithms to a set of block-structured process models with clearly specified conditions regarding the position of their interdependent nodes.

The assumption that each business process only exists in one BPA subset may be too strict in practice. A supporting business process may be related to many core business processes of a company, e. g., the inbox handling process, that handles all incoming request of services and orders and distributes them to the according core processes. Applying our BPA in such cases without treating support processes differently would result into a very large BPA subset. This BPA subset, starting from the inbox handling process, would branch out into all core processes that require the inbox handling process as input. Such a scenario illustrates the advantages and disadvantages of our approach. On the one hand one can see the importance of that inbox handling process. If it fails all other processes will halt. On the other hand, it results into a complex overview on all interdependent processes. In this case, a two stage approach would be favorable where such supporting processes can be replicated in each BPA subset and an aggregation of those subsets along the supporting process is provided as well.

In Chapter 5 we introduced structural and behavioral properties for BPAs. Those properties cover basic aspects of a BPA structure and the characteristics of the inherent business processes. The behavioral properties aim at finding out if the BPA under examination has errors and if there exist BPA runs that reach the desired final state. It would be desirable to identify desired paths through the BPA. In many cases there are different paths to reach the desired final state. A terminating BPA run may describe a process execution chain which contains the participation of undesired processes for a particular case. To identify the desired terminating BPA runs domain knowledge is necessary as our approach assumes all terminating runs of a BPA to be favorable. It cannot automatically identify if the terminating runs include the desired paths and desired processes without input of an expert with domain knowledge.

Part of the presented issue could be resolved by introducing correlation techniques. So far, our BPA approach does not provide correlation techniques to route cases through a BPA. Hence we cannot distinguish several cases in one BPA run and assume one BPA run depicts one overall case of a scenario. However, in reality one process may collect information from several cases and then process them. Similar, one process may distribute several cases to different processes that interact with each other and result in a terminating BPA run which under consideration of correlation would result into a deadlock.

Nonetheless, our approach is the first to introduce structural and behavioral properties on BPA level. We provide a solid foundation that can be extended by further structural and behavioral properties if required.

BPA ANALYSIS. The pattern based analysis technique presented in Section 6.1 is applicable to examine the direct relationship of two processes. Indirect relationships between processes are not considered. Hence, a BPA that does not contain any anti-pattern may still have a deadlock or livelock, or a blocking situation caused by indirect interdependencies. Despite its limitation the presented technique provides a first step for analysis and improving the quality of a BPA. The pattern approach can be applied to synchronous and asynchronous environments. Many use cases assume asynchronous environments and hence many techniques only focus on asynchronous communication as the techniques on process model level described in [156, 78, 92, 28]. Synchronous communication has not been looked at in many approaches. Decker and Weske [28] assume that the message exchange between two process happens synchronously. In this case both processes that exchange a message are both ready when the message exchange is performed according to the modeled choreography.

The pattern and anti-pattern approach highlights the possibility that in some cases this assumption may not hold if the receiving process has not been started before the message is sent. Then it is not ready for receiving the message. The anti-pattern analysis relies on a causal relation showing that there is no possibility that the processes will be active at the time of sending, e. g., the sender both triggers and sends a message to the partner process at the same time. The assumption of a synchronous communication environment mainly applies to real-time processes where communication cannot be buffered. The message is hence either read or lost as it looses its value instantly.

The BPA analysis based on an ON transformation provides more powerful analysis of BPAs and extends the pattern and anti-pattern based approach. Despite that, it can only be applied for asynchronous communication exchanges in which information flows and triggers are buffered until the process is ready to read them. The use of ONs for the analysis of BPAs has many advantages as different analysis tools exist. However, it is accompanied by the known disadvantages of state space explosion and undecidability of specific model checking problems. This is the case for BPAs with high use of large multiplicity sets or BPAs with many business processes. Nonetheless, the scope of a BPA scenario and the fact that BPAs contain only abstract models reduces the state space for the analysis. In general, we expect a BPA subset to consist of a manageable set of business processes. From our experience with partners from the financial and public sector we observed BPA subsets containing 15-25 business processes. However, this varies in regard to organization, domain, and modeling guidelines followed. The average size of BPAs would need to be examined by a larger empirical survey.

Currently each terminating BPA run by default is considered to represent a desired path. In this regard our analysis technique cannot verify if all participating processes in the BPA run are supposed to participate in that BPA run. However, our ON based analysis technique provides the foundation for the development of such a functionality. A user, the domain expert, could select a range of processes from the BPA through a graphical interface. Based on the selection the analysis could be extended to check if the terminating runs involve the desired paths and processes by extending the according CTL formulae. Similarly, this could be solved by introducing a correlation mechanism but also here domain knowledge is necessary to specify which processes should partake in one run and how they should interact. Our BPA approach is the first to introduce analysis techniques on the BPA level in regard to structural and behavioral aspects. Those serve as foundation for analysis and can be extended, if required. For instance, enriching BPA runs with additional process meta-data like costs, (wait) time, or human resources required leverages process information to an end-to-end process examination.

BPA METHODOLOGY. Our BPA methodology describes the application of our BPA approach and its additional techniques in a BPM project. It describes a bottom-up BPA extraction and a top-down decomposition algorithm that support ensuring consistency between BPA level and process model level.

The BPA extraction algorithm covers structural sound block-structured process models. However, the extraction algorithm is limited in the extraction of XOR-bonds. XOR-bonds that contain events of different types (i. e., sending and receiving events) cannot be extracted with the current configuration. Despite that, the evaluation of process model language constructs used in BPMN and EPC process models of the BPMAI model collection by Kunze et al. [71] indicates a good coverage of the presented extraction algorithms. Kunze et al. [71] assume that most models describe sequential behavior as only about 50% of the process models use XOR-gateways and about 40% use AND-gateway. An extensive evaluation of several process model collections to indicate the exact characteristics and the position of sending and receiving nodes in process models is necessary to extend and adapt our extraction algorithm in future work.

The extraction algorithm could be configured to address extraction of XOR-bonds containing mixed node types with the consequence of loosing information and inferring more behavior on BPA level, i. e., the BPA looses accuracy, as depicted by the modified extraction algorithm of Breske [15] in Section 7.2.6. Another option is an extension of the current BPA constructs with complex events, e. g., that depict bi-directional message exchange in one symbol, i. e., in one BPA run process p1 sends a message to process p2 and in another run p2 sends a message to p1. By introducing new event types, our BPA approach would be applicable to an even larger set of process models.

Our BPA methodology supports the creation of a BPA bottom-up and developing process models top-down. While not specifying business process identification guidelines and guidelines on how to cut business processes, the provided overview, visualization, and analysis of business process interdependencies support that process. A BPA shows the overall picture of the process interdependencies. It allows reflecting on those interdependencies and supports the (re-)design a process model collection. For example, the observations made on BPA level may lead to the fusion of two or more processes or the division of a process into smaller ones.

10.3 FUTURE RESEARCH

In this thesis we presented the design of a novel BPA approach that allows specifying and analyzing horizontal interdependencies and multi-communication between processes. We have developed a novel BPA approach with a set of analysis and extraction techniques that support the re-/design and maintenance of process models in a process model repository by addressing quality issues like correctness, consistency, and harmonized process interdependencies. While our BPA approach provides a complete tool set for structuring, managing, and analyzing business processes within a process model collection, this thesis is just a starting point for future research on PAs.

The automatic identification and visualization of relevant and irrelevant BPA runs from the various possibilities that a BPA offers, would provide great usability to the end user. So far a BPA may produce a variety of terminating runs of which not all may be relevant. We sketched an initial idea based on input from domain experts but ideally such information would be automatically extracted from the business process models or a related business model. The implementation of a correlation mechanism should be investigated in this regard in the future. For instance, cases with specific characteristics must take a particular path through the BPA defined by a correlation technique.

Strategic planning and orientation in companies is rather based on business models that are independent from the process model information despite the fact that business processes convey the business operations and execution logs provide valuable information. Business model, PA level, and process model level are not connected. So far we proposed a technique to provide consistency between BPA level and process model level in our BPA methodology. We did not consider the linkage to the business model level. This area requires further research in regard to which information should be provided in process models, in BPA subsets, and in a BPA compendium to derive a business model and vice versa. In this way the use of business process information for the strategic planning could be facilitated for business managers. As first step in this direction, it would be desirable to provide a BPA extraction that only shows the core interdependencies between business processes to reduce the complexity of interdependencies to one type of relation between two processes. The data relations defined for the aggregation of data dependencies in Chapter 8 could serve as starting point for the identification and development of complex relation types and interdependency patterns.

By straightjacking processes into a sequential processes on BPA level we create simple overview on complex interdependencies but in the same time restrict the complexity of process models that our BPA extraction algorithm covers. The extensions of our BPA approach with elements that allow depicting the complex interdependency patterns in a simple way is an evident next step. Consequently, highly aggregated BPA overviews could be generated and the BPA approach could be extended covering an even larger set of process models. This requires empirical research which examines the use of our BPA approach in more detail and the identification of interdependency patterns used in process models in the future.

A APPENDIX

The following table summarizes related work and categorizes it in three major research areas as presented in Chapter 3. These three research areas are business process architecture approaches, choreography approaches on model level, and business process model abstraction. The table is based on the major characteristics used for evaluating and classifying PA approaches that we identified in PA surveys and PA evaluation frameworks. These aspects are: aim of methodology, framework characteristics, scope of the approach, type of relationship and interdependencies depicted in the framework, and the elements involved. The different aspects used in Table 7 are described in more detail in Section 3.5.

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
Enterprise Architectures	Zachman [174]	Overview, Visualization, Classification	Informal, Top-Down, Relationship between Domains	Single	Multi-faceted	Input/Output	Process, Business Object (Data), Actors
	Scheer [128]	Overview, Visualization, Classification	Informal, Top-Down, Bottom-Up, Relationship between Layers	Single	Consists of/ Relates to	(Relates to)	Function, Reference
Business Process Architectures	Scheer et al. [129, 127, 22]	Overview, Visualization, Classification	Informal, Top-Down, Bottom-Up, Relationship between Layers	Single	Consists of/ Relates to	(Relates to)	Function, Reference
	Koliadis et al. [70]	Analysis, Evaluation Framework	Formal, Top-Down, Capability Model Relationship with Process Model, based on I* and TROPOS	Single	Process realizes Service Outcome or Capability	-	Goal, Actor, Process, Capability (direction EA)
	Dijkman et al. [35, 31]	Overview, Classification, Structure and Manage, Identification, Design	Informal, Top-Down, Several Views, Guideline Support	Single	Multi-faceted, Aggregation, Specialization	-	Function, Business Objects
	Schmelzer and Sesselmann [130]	Structure and Manage, Overview, Classification	Informal, Bottom-Up, Top-Down	Single, (Multiple (on Landscape Level)	Decomposition	Requirement, Service Delivery	Function, Process (on Landscape Level)
	Andersson et al. [2]	Re-/Design, Classification	Informal, Top-Down	Single	-	-	Goal, Reference, Patterns
	Harmon [58, 59]	Classification, Measurement, Decision Making Support	Informal, Top-Down, Practice	Single	Aggregation	-	Function, Goals
	Muehlen et al. [106]	Classification, Structure and Manage, Re-/Design, Optimization, Reduction of Complexity	Informal, four Levels, Guidelines	Single	Aggregation, Decomposition	Predecessor/ Successor (Value Chains, End-to-End)	Function, Department (Actor)
	vom Brocke and Rosemann [16, 17]	Classification, Structure and Manage	Informal, Top-Down	Single	Aggregation, Decomposition	-	Function, Action, Reference

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
	Malone et al. [86]	Classification, Structure	Informal, Top-Down, Re-Use, Knowledge-base	Single	Parts of, Uses, Specailization, Generalization	-	Function, Reference
	Jung et al. [64]	Classification, Overview, Structure and Manage	Formal (Cluster Algorithm), Bottom-Up, Automation	Single	Hierarchical, Similarity	-	Action
	APQC [3]	Classification, Standardization	Informal, Top-Down, Domain Specific	Single	Hierarchical	-	Function, Business Objects, Reference
	SCOR [133]	Classification, Standardization, Re-/Design, Optimization	Informal, Top-Down, Reference Model	Single	Hierarchical	-	Function, Goal, Reference
	Castellanos and Correal [18]	Classification, Alignment with EA	Informal	Single	Alignment with Information Systems	-	Business Object (Data)
	Klein and Petti [67]	Re-/Design, Process Model Generation	Informal, Efficient re-/design	Single	Hierarchical	-	Labels
	Moreira et al.[105]	Classification	Informal	Single, (Multiple)	Part of	Message Exchange	Business Objects (Product)
	Jacobs et al. [63]	Re-/Design, Overview of different aspects, Visualization	Informal, Automation, Reference	Multiple	Consists of, enabled by	n.d.	Reference Models, Function, Activity
	Maddern et al. [83]	Overview, Re-/Design	Survey	Multiple		Predecessor/ Successor (end-to-end)	Process
	Rulle and Siegeris [126]	Execution, Configuration	Formal, Application of our Pattern and Anti-pattern Approach	Multiple	-	Trigger, Message Exchange	Business Objects (Data, State centric)

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
	Lind and Goldkuhl [76]	Business Modeling	Informal, Patterns,Consistency, Top-Down	Single, Multiple	Hierarchical	Business Act, Action Pair, Exchange, Business Transaction, Transaction group	Action, Actor
	Green et al. [57, 56, 55]	Re-Use, Overview, Visualization	Informal, Top-Down	Single, Multiple	Encapsulates	Activates/ Interacts with	Department and Roles (Actors), Process
Navigation and Visualization (BPA)	Srivastava and Mukherjee [142]	Classification	Formal, Bottom-Up, Automation	Single	Similarity	-	Business Object, Data, Process Elements
	Melcher and Seese [96]	Classification, Visualization, Navigation	Formal, Bottom-Up, Automation	Single	Similarity, K-Means Cluster	-	Process Metrics, Process
	Hipp et al. [60]	Visualization, Navigation, Reduction of Complexity	Informal	Single	Hierachical		Process
	Milde [104]	Visualization, Navigation, Re-/Design	Informal	Single			Process
	Luebbe and Schnaegelberger [81]	Classification	Informal, Survey, Practice	Single	Hierarchical		Function, Object, Reference, Goal, Department
BPMA and Process Model Configuration/Customization	Polyvyanyy et al. [116, 115, 117]	Abstraction, Reduction of Complexity, Overview	Formal, Bottom-Up, Comprehensibility	Single	Aggregation/ Decomposition	-	Activities
	Smirnov et al. [135, 140, 137, 138, 136, 141]	Abstraction, Reduction of Complexity, Overview, Visualization	Formal, Bottom-Up, Comprehensibility	Single	Aggregation/ Decomposition	-	Activities
	Reijers et al. [122]	Reduction of Complexity, Overview	Formal, Comprehensibility	Single	Aggregation, Hierarchical	-	(Sub-) Processes
	Reijers et al. [123]	Classification, Visualization, Management, Maintenance	Informal, Comprehensibility	Single	Aggregation, Generalization/ Specialization	-	Business Object (Product)

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
	Eshuis and Grefen [47]	Aggregation, Reduction of Complexity, Overview, Visualization	Formal, Bottom-Up	Single	Hierarchical	-	Activities
	Reichert et al. [121]	Aggregation, Reduction of Complexity, Overview, Visualization (personalized)	Formal, Bottom-Up, Consistency	Single	Aggregation, Child/Parent	-	Activities
	Baran et al. [8]	Classification, Configuration, Re-/Use, Reduction of Complexity, Abstraction	Formal	Single	Encapsulates	-	Process, Sub-processes
Business Process Choreographies	Barros et al. [10]	Model Inter-Organizational Interaction, Issues and Solutions for Modeling Interaction	Informal, Interaction Patterns	Multiple	-	Message Exchange (one-to-many)	Message, Activities
	Aalst et al. [155]	Model Inter-Organizational Interaction, Analysis, Composition and Replacement of Services	Formal, Composition Properties (strategy, controllability, and accordance, controllability), Pattern and Anti-Patterns	Multiple	-	Message Exchange (one-to-many)	Message, Activities
	van Glabbeek and Stork [158]	Model Inter-Organizational Interaction, Analysis	Formal, Composition, Global Termination, Query nets	Multiple	Process and sub-process	Procedure Call, Message Exchange (one-to-one)	Process, Activities
	Martens [90, 91]	Model Inter-Organizational Interaction, Analysis, Composition and Replacement of Services	Formal, Composition Properties (usability, compatibility, equivalence), Correctness properties (absence of deadlocks, soundness)	Multiple	-	Message Exchange (one-to-one)	Process, Activities
	Decker [24], Decker and Weske [27, 28]	Model Inter-Organizational Interaction, Analysis	Formal, Interaction Properties (local enforceability, realizability)	Multiple	-	Message Exchange	Role, Actor, Activities

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
	Massuthe [92], Massuthe and Schmidt [93], Lohmann et al. [79]	Composition of Services, Analysis, Inter-Organizational Interaction	Formal, Composition, Interaction Properties (controllability, strategy)	Multiple	-	Message Exchange	Service, Process, Activities
	Baldan et al. [7]	Model Inter-Organizational Interaction, Composition	Formal, Composition	Multiple	-	Message Exchange	Service, Process, Activities
	Lohmann [78]	Composition of Services, Analysis, Inter-Organizational Interaction	Formal, Composition, Interaction Properties (controllability, strategy), Correctness	Multiple	-	Message Exchange	Service, Process, Activities
	Aalst et al. [151, 152, 154], Mans and Russel[87], Mans et al. [89] , Mans [88]	Execution, Reduction of Complexity, Model Interaction on Instance Level	Formal, Flexibility, Instance Level	Single, (Multiple)	-	Message Exchange (one-to-many, many-to-many)	Activities, Process
Data in Business Process Choreographies	Mueller et al. [108]	Flexibility, Re-/Design, Maintenance of data objects, Consistency between Data and Control Flow Model, Re-Use on Process Model and Instance level	Formal, Top-Down	Single	Hierarchical	-	Business Objects
	Kuenzle and Reichert [73]	Flexibility, Re-/Design, Maintenance of data objects, Consistency between Data and Control Flow Model, Re-Use of Process Model and Instance level	Informal, Top-Down, two Granularity Levels	Single	Hierarchical (data)	-	Busines Objects, Function, Process
	Meyer and Weske [102]	Consistency between Data OLC and Process Model	Formal, Consistency, Conformance, OLC	Single	-	Read / Write Access	Activites, Business Object (Data)

Research Area	Source	Aim	Framework Characteristics	Scope	Relationship type		Elements of Focus
					Hierarchical	Horizontal	
	Knuplesch et al.[69, 68]	Analysis, Specification of Inter-Organizational Data Exchange	Formal, Correctness (termination and realizability)	-	Multiple	Message Exchange	Data, Message Tasks, Actor
	Fahland et al. [51]	Analysis	Formal, Proclets, Process Instance Level and Process Model Level, Conformance	Multiple	Decomposition	Proclet Interaction (one-to-many, many-to-many)	Data, Processes (Activity)
	Meyer et al. [103]	Synchronization of OLCs for one Process Model, Execution	Formal, Automation, Conformance	Multiple	-	Message Exchange	Business Object (Data), Activities

Table 7: Classification of Related Work

GLOSSARY

PM	Process model, see Definition 1, page 18
N	The set of nodes of a process model, consisting of activities, gateways, events. See Definition 1, page 18
A	The set of activities of a process model, see Definition 1, page 18
E	The set of events of a process model, see Definition 1, page 18
G	The set of gateways of a process model, see Definition 1, page 18
$\mathcal{D}$	The set of data nodes of a process model Definition 1, page 18
CF	The control flow of a process model, see Definition 1, page 18
DF	The data flow between two nodes being either activity or event, see Definition 1, page 18
type	A function that assigns a type to a gateway, see Definition 1, page 18
OLC	The object life cycle of a data object, see Definition 3, page 19
S	The set of data object states, see Definition 3, page 19
i	The initial state of a data object in the OLC, see Definition 3, page 19
S_F	The set of final states of a data object in the OLC, see Definition 3, page 19
T_D	An acyclic relation of data state transitions, see Definition 3, page 19
Σ	A finite set of actions for manipulating the corresponding data object, see Definition 3, page 19
$SESE$	Single entry/ single exit fragment of a process model, see page 20, page 20
nPM	A normalized process model with a single start and a single end node, see Definition 4, page 20
s	The only start node of a process model, see Definition 4, page 20
f	The only end node of a process model, see Definition 4, page 20
$\mathcal{G}$	A graph, see Definition 5, page 21
$\mathcal{E}$	The set of edges of a graph, see Definition 5, page 21

$\mathcal{N}$	The set of nodes of a graph, see Definition 5, page 21
PMF	A process model fragment, see Definition 6, page 21
Δ	The set of all canonical components of a process model, see Definition 8, page 21
$RPST_{PM}$	The refined process structure tree of a process model PM, see Definition 10, page 22
r	The root node of an $RPST_{PM}$ tree of a process model, see Definition 10, page 22
ξ	Hierarchical parent child relation between component and its child in a process model, see Definition 10, page 22
TR	A trivial component of a process model, see page 22, page 22
PO	Polygon, depicts a sequence of nodes or a sequence of components, see page 22, page 22
BO	A bond consists of components that share common boundary nodes, see page 22, page 22
RI	Rigid, a component of a process model that cannot be classified into trivial, polygon, or bond, see page 22, page 22
ct	The function *component type* assigns a type (trivial, polygon, bond, or rigid) to a component of a process model, see Definition 9, page 22
PN	Petri net, a state transition system, see Definition 11, page 29
P	The set of places of a Petri net, see Definition 11, page 29
T	The set of transitions of a Petri net, see Definition 11, page 29
F	The flow relation of a Petri net, see Definition 11, page 29
M	The marking of a Petri net, see Definition 11, page 29
M_s	The initial marking of a Petri net, see Definition 11, page 29
PNS	A Petri net system, depicted by a Petri net with designated initial marking, see Definition 11, page 29
ω	Weighting function that assigns a weight to an arc, see Definition 13, page 30
$WFnet$	Workflow net, a subclass of Petri nets with structural restrictions, see Definition 18, page 31
M_Ω	The final marking of a Workflow net, see Definition 19, page 31
p_i	The initial place of a Workflow net, see Definition 18, page 31

p_o	The final place of a Workflow net, see Definition 18, page 31
$\mathcal{O}$	Open net, a subclass of Petri nets with designated interface places and final markings, see Definition 23, page 33
Ω	The final marking of an Open net, see Definition 23, page 33
$\mathcal{E}$	The set of events of a business process architecture, see Definition 29, page 76
e	An event of a process in a business process architecture, see Definition 29, page 76
$\mathcal{V}$	The set of processes of a business process architecture, see Definition 29, page 76
v	A process of a business process architecture, see Definition 29, page 76
$\mathcal{L}$	The message flow relation between two events in a business process architecture, see Definition 29, page 76
$\mathcal{I}$	The trigger relation between two events in a business process architecture, see Definition 29, page 76
BPA	A Business Process Architecture, see Definition 29, page 76
χ	The conflict relation between flows of a BPA, indicating flows that are mutually exclusive, see Definition 29, page 76
μ	Denotes the multiplicity set of an event, see Definition 29, page 76
$\pm$	The correspondence relation between two events of the same process, demanding that they send, respectively receive, the same number of messages, see Definition 29, page 76
PMC	A process model collection that consists of a set of process models and their message flows, see Definition 43, page 147
MF	The message flow relation between two process models in a process model collection, see Definition 43, page 147
K	A subset of nodes being events and activities that take part in an interaction between two process models, see Definition 43, page 147
$\mathbb{F}$	Overall set of flows of a process model collection, see Definition 43, page 147
$mapflow$	A function that maps message flows to BPA information flow and trigger, see Definition 45, page 151

ret	A function retrieves for a node the matching event in a BPA, see Definition 45, page 151
PMS	A process model skeleton depicts the nodes that are part of a message flow, see Definition 47, page 166

BIBLIOGRAPHY

[1] Rama Akkiraju and Anca Ivan. Discovering Business Process Similarities: An Empirical Study with SAP Best Practice Business Processes. In *Service-Oriented Computing - ICSOC 2010*, volume 6470 of *LNCS*, pages 515–526. Springer, 2010.

[2] Birger Andersson, Ilia Bider, Paul Johannesson, and Erik Perjons. Towards a formal definition of goal-oriented business process patterns. *Business Process Management Journal*, 11(6):650–662, 2005. ISSN 1463-7154. doi: 10.1108/14637150510630846.

[3] APQC. APQC - Process Classification Framework, 2014. URL `http://www.apqc.org/knowledge-base/download/313690/K05162_PCF_Ver_61_1.pdf`.

[4] Ahmed Awad. BPMN-Q: A Language to Query Business Processes. In *EMISA*, pages 115–128, 2007.

[5] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999. ISBN 020139829X.

[6] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[7] Paolo Baldan, Andrea Corradini, Hartmut Ehrig, and R Heckel. Compositional Modeling of Reactive Systems Using Open Nets. In KimG. Larsen and Mogens Nielsen, editors, *CONCUR 2001 - Concurrency Theory*, volume 2154 of *LNCS*, pages 502–518. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42497-0. doi: 10.1007/3-540-44685-0_34.

[8] M Baran, K Kluza, G J Nalepa, and A Ligeza. A hierarchical approach for configuring business processes. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 915–921, September 2013.

[9] Alistair Barros, Egon Börger, Kung-Kiu Lau, and Richard Banach. A Compositional Framework for Service Interaction Patterns and Interaction Flows. *Formal Methods and Software Engineering*, 3785:5–35, 2005. doi: 10.1007/11576280.

[10] Alistair Barros, Marlon Dumas, Arthur ter Hofstede, W. M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera. Service Interaction Patterns. In Wil M. P. Aalst,

Boualem Benatallah, Fabio Casati, and Francisco Curbera, editors, *BPM*, volume 3649 of *LNCS*, pages 302–318, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-28238-9. doi: 10.1007/11538394.

[11] Jörg Becker, Lars Algermissen, T Falk, and Inc Ebrary. *Prozessorientierte Verwaltungsmodernisierung: Prozessmanagement im Zeitalter von E-Government und New Public Management*. Springer, 2007.

[12] Catriel Beeri, Anat Eyal, Simon Kamenkovich, and Tova Milo. Querying Business Processes. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 343–354. VLDB Endowment, 2006.

[13] Philip A Bernstein and Umeshwar Dayal. An Overview of Repository Technology. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 705–713, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

[14] David Bovet and Joseph Martha. Value nets: reinventing the rusty supply chain for competitive advantage. *Strategy & Leadership*, 28(4):21–26, 2000. doi: 10.1108/10878570010378654.

[15] Robert Breske. *Business Process Architecture Extraction with regard to Inter-Processs Dependencies (Master Thesis)*. Master thesis, University of Potsdam, 2014.

[16] Jan Vom Brocke and Michael Rosemann. *Handbook on Business Process Management 1*. Springer Berlin Heidelberg, 2010. ISBN ISBN 978-3-642-45099-0.

[17] Jan Vom Brocke and Michael Rosemann. *Handbook on Business Process Management 2*. Springer Heidelberg, 2010.

[18] Camilo Castellanos and Dario Correal. KALCAS: A FrameworK for Semi-automatic ALignment of Data and Business ProCesses ArchitectureS. In Tadeusz Morzy, Theo Härder, and Robert Wrembel, editors, *Advances in Databases and Information Systems*, volume 7503 of *LNCS*, pages 111–124. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33073-5. doi: 10.1007/978-3-642-33074-2_9.

[19] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press Cambridge, 1999. ISBN 9780262032704.

[20] David Cohn and Richard Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin*, 32(3):3–9, 2009.

[21] T A Curran and G Keller. *SAP R/3 Business Blueprint - Business Engineering mit den R/3-Referenzprozessen*. Addison-Wesley, Bonn, Germany, 1999.

[22] Rob Davis. *ARIS Design Platform: Advanced Process Modelling and Administration*. Springer Berlin Heidelberg, 2008. ISBN 9781848001114.

[23] Gero Decker. Realizability of interaction models. In *ZEUS*, pages 55–60. Citeseer, 2009.

[24] Gero Decker. *Design and analysis of process choreographies*. PhD thesis, University of Potsdam, 2009. URL `http://opus.kobv.de/ubp/volltexte/2010/4076/`.

[25] Gero Decker and Jan Mendling. Process instantiation. *Data & Knowledge Engineering*, 68(9):777–792, 2009. ISSN 0169-023X. doi: 10.1016/j.datak.2009.02.013.

[26] Gero Decker and Mathias Weske. Behavioral consistency for B2B process integration. In *Proceedings of the 19th international conference on Advanced information systems engineering*, CAiSE'07, pages 81–95, Berlin, Heidelberg, June 2007. Springer-Verlag. ISBN 978-3-540-72987-7.

[27] Gero Decker and Mathias Weske. Local Enforceability in Interaction Petri Nets. *Business Process Management*, pages 305–319, 2007.

[28] Gero Decker and Mathias Weske. Interaction-centric modeling of process choreographies. *Information Systems*, 36(2):292–312, 2011. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2010.06.005.

[29] Gero Decker, Oliver Kopp, and Frank Leymann. Modeling service choreographies using BPMN and BPEL4Chor. In *Advanced Information Systems Engineering*, volume 5074 of *LNCS*, pages 79–93. Springer Berlin Heidelberg, 2008.

[30] Juliane Dehnert and Peter Rittgen. Relaxed Soundness of Business Processes. In KlausR. Dittrich, Andreas Geppert, and MoiraC. Norrie, editors, *Advanced Information Systems Engineering*, volume 2068 of *LNCS*, pages 157–170. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42215-0. doi: 10.1007/3-540-45341-5_11.

[31] Remco M. Dijkman. Designing a Process Architecture - A Concrete Approach. Technical report, TU Eindhoven.

[32] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Inf.*

Softw. Technol., 50(12):1281–1294, November 2008. ISSN 0950-5849. doi: 10.1016/j.infsof.2008.02.006.

[33] Remco M. Dijkman, I. Vanderfeesten, and Hajo A. Reijers. The Road to a Business Process Architecture: An Overview of Approaches and their Use. 2011. URL `http://cms.ieis.tue.nl/Beta/Files/WorkingPapers/wp_350.pdf`.

[34] Remco M. Dijkman, Marcello La Rosa, Hajo A. Reijers, and Marcello La Rosa. Managing large collections of business process models - Current techniques and challenges. *Computers in Industry*, 63(2):91–97, 2012.

[35] Remco M. Dijkman, Irene Vanderfeesten, and Hajo A Reijers. Business process architectures: overview, comparison and framework. *Enterprise Information Systems*, pages 1–30, 2014. doi: 10.1080/17517575.2014.928951.

[36] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-33142-8. doi: 10.1007/978-3-642-33143-5.

[37] Rami-Habib Eid-Sabbagh. Towards Automatic Generation of Process Architectures for Process Collections. In *4th Central-European Workshop on Services and their Composition (ZEUS 2012) On-site Proceedings*, pages 88–95, 2012.

[38] Rami-Habib Eid-Sabbagh and Mathias Weske. Analyzing Business Process Architectures. In Camille Salinesi, MoiraC. Norrie, and Óscar Pastor, editors, *Advanced Information Systems Engineering*, volume 7908, pages 208–223. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-38709-8_14.

[39] Rami-Habib Eid-Sabbagh and Mathias Weske. From Process Models to Business Process Architectures: Connecting the Layers. In *9th. International Workshop on Engineering Service-Oriented Applications (WESOA 13)*, pages 4–15, 2013.

[40] Rami-Habib Eid-Sabbagh, Remco M. Dijkman, and Mathias Weske. Business Process Architecture: Use and Correctness. In Alistair P Barros, Avigdor Gal, and Ekkart Kindler, editors, *BPM*, volume 7481 of *LNCS*, pages 65–81. Springer, 2012. ISBN 978-3-642-32884-8.

[41] Rami-Habib Eid-Sabbagh, Matthias Kunze, Andreas Meyer, and Mathias Weske. A Platform for Research on Process Model Collections. In *BPMN2012 Workshop proceedings*, 2012.

[42] Rami-Habib Eid-Sabbagh, Matthias Kunze, and Mathias Weske. An Open Process Model Library. In Florian Daniel, Kamel

Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops (PMC2011)*, volume 100 of *LNBIP*, pages 26–38, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-28114-3. doi: 10.1007/978-3-642-28115-0.

[43] Rami-Habib Eid-Sabbagh, Marcin Hewelt, Andreas Meyer, and Mathias Weske. Deriving Business Process Data Architectures from Process Model Collections. In Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors, *11th International Conference on Service-Oriented Computing*. Springer Berlin Heidelberg, 2013.

[44] Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. Business Process Architectures with Multiplicities : Transformation and Correctness - Technical Report. Technical report, Hasso Plattner Institute, University of Potsdam, Potsdam, 2013.

[45] Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. Business Process Architectures with Multiplicities: Transformation and Correctness. In *BPM*, volume 8094 of *LNCS*, pages 227–234. Springer, 2013. ISBN 978-3-642-40175-6. doi: 10.1007/978-3-642-40176-3_19.

[46] Rami-Habib Eid-Sabbagh, Marcin Hewelt, and Mathias Weske. A Tool for Business Process Architecture Analysis. In Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors, *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013*. Springer Berlin Heidelberg, 2013. doi: 10.1007/978-3-642-45005-1_61.

[47] Rik Eshuis and Paul Grefen. Constructing customized process views. *Data & Knowledge Engineering*, 64(2):419–438, February 2008. ISSN 0169023X. doi: 10.1016/j.datak.2007.07.003.

[48] Rik Eshuis and Peter van Gorp. Synthesizing Object Life Cycles from Business Process Models. In *Conceptual Modeling*, pages 307–320. Springer, 2012.

[49] Javier Esparza. Decidability and Complexity of Petri Net Problems. *Petri Nets: Fundamental Models, Verification and Applications*, pages 87–122, 1998. doi: 10.1002/9780470611647.ch4.

[50] Javier Esparza and Mogens Nielsen. Decidability Issues for Petri nets. *Bulletin of the EATCS*, 52(May):244–262, 1994. doi: 10.1.1.16.5995.

[51] Dirk Fahland, Massimiliano De Leoni, Boudewijn F. Van Dongen, and Wil M. P. van der Aalst. Conformance Checking of Interacting Processes with Overlapping Instances. In *Proceedings of the 9th International Conference on Business Process*

Management, BPM'11, pages 345–361, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23058-5.

[52] Dirk Fahland, Cedric Favre, and Jana Koehler. Analysis on demand: Instantaneous soundness checking of industrial business process models. *Data & Knowledge Engineering*, 70(5):448–466, 2011.

[53] Peter Fettke and Peter Loos. Classification of reference models: a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, January 2003. ISSN 1617-9846. doi: 10.1007/BF02683509.

[54] Peter Fettke, Peter Loos, and Jörg Zwicker. Business Process Reference Models: Survey and Classification. In ChristophJ. Bussler and Armin Haller, editors, *Business Process Management Workshops*, volume 3812 of *LNCS*, pages 469–483. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-32595-6. doi: 10.1007/11678564_44.

[55] Stewart Green and Martyn Ould. The Primacy of Process Architecture. In *CAiSE Workshops (2)*, pages 154–159, 2004.

[56] Stewart Green and Martyn Ould. A framework for classifying and evaluating process architecture methods. *Software Process: Improvement and Practice*, 10(4):415–425, October 2005. ISSN 1077-4866. doi: 10.1002/spip.244.

[57] Stewart Green, Ian Beeson, and Richard Kamm. Process Architectures and Process Models : Opportunities for Reuse. In *8th Workshop on Business Process Modeling, Development, and Support*, 2007.

[58] Paul Harmon. Using Balanced Scorecard to Support a Business Process Architecture. *BPTrends*, 5(17):3, 2007.

[59] Paul Harmon. *Business process change: A guide for business managers and BPM and Six Sigma professionals*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2007. ISBN 9780123741523.

[60] Markus Hipp, Bela Mutschler, and Manfred Reichert. Navigating in Process Model Collections: A New Approach Inspired by Google Earth. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops*, volume 100 of *LNBIP*, pages 87–98. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-28114-3. doi: 10.1007/978-3-642-28115-0_9.

[61] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns – Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 0321200683.

[62] Constantin Houy, Peter Fettke, Peter Loos, W. M. P. van der Aalst, and John Krogstie. BPM-in-the-Large - Towards a Higher Level of Abstraction in Business Process Management. In Marijn Janssen, Winfried Lamersdorf, Jan Pries-Heje, and Michael Rosemann, editors, *E-Government, E-Services and Global Processes*, volume 334 of *IFIP Advances in Information and Communication Technology*, pages 233–244. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15345-7. doi: 10.1007/978-3-642-15346-4_19.

[63] Dina Jacobs, Paula Kotzé, and Alta Van Der Merwe. Towards an enterprise repository framework. In *1st International Workshop on Advanced Enterprise Repositories*, pages 77–89, 2009.

[64] Jae-yoon Jung, Joonsoo Bae, and Ling Liu. Hierarchical Business Process Clustering. *2008 IEEE International Conference on Services Computing*, 2:613–616, 2008. doi: 10.1109/SCC.2008.69.

[65] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, May 1969. ISSN 00220000. doi: 10.1016/S0022-0000(69)80011-5.

[66] G Keller, M Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)", 1992.

[67] Mark Klein and Claudio Petti. A handbook-based methodology for redesigning business processes. *Knowledge and Process Management*, 13(2):108–119, 2006. ISSN 1099-1441. doi: 10.1002/kpm.248.

[68] David Knuplesch, Rüdiger Pryss, and Manfred Reichert. A Formal Framework for Data-Aware Process Interaction Models. Technical Report UIB-2012-06, University of Ulm, Ulm, 2012. URL http://dbis.eprints.uni-ulm.de/860/.

[69] David Knuplesch, Rüdiger Pryss, and Manfred Reichert. Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 223–232, 2012.

[70] George Koliadis, Aditya K Ghose, and Srinivas Padmanabhuni. Towards an Enterprise Business Process Architecture Standard. *Services, IEEE Congress on*, 0:239–246, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/SERVICES-1.2008.60.

[71] Matthias Kunze, Alexander Luebbe, Matthias Weidlich, and Mathias Weske. Towards Understanding Process Modeling - The Case of the BPM Academic Initiative. In Remco M Dijkman, Jörg Hofstetter, and Jana Koehler, editors, *BPMN*, volume 95 of *LNBIP*, pages 44–58. Springer, 2011. ISBN 978-3-642-25159-7.

[72] Matthias Kunze, Philipp Berger, and Mathias Weske. BPM Academic Initiative – Fostering Empirical Research. In *Demo Sessions of the 10th International Conference on Business Process Management*, 2012.

[73] Vera Künzle and Manfred Reichert. PHILharmonicFlows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):205–244, 2011. ISSN 1532-0618. doi: 10.1002/smr.524.

[74] Marcello La Rosa, H M Arthur, Lijie Efficient, Tao Jin, Jianmin Wang, La Rosa, and Marcello La. Efficient and Accurate Retrieval of Business Process Models through Indexing. *Computers in Industry*, 2010.

[75] Marcello La Rosa, Hajo A. Reijers, W. M. P. van der Aalst, Remco M. Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. APROMORE: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, June 2011. ISSN 09574174. doi: 10.1016/j.eswa.2010.12.012.

[76] Mikael Lind and Göran Goldkuhl. Generic Layered Patterns for Business Modelling. In *Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling*, 2001.

[77] Rong Liu, Frederick Y Wu, and Santhosh Kumaran. Transforming Activity-Centric Business Process Models into Information-Centric Models for SOA Solutions. *J. Database Manag.*, 21(4): 14–34, 2010.

[78] Niels Lohmann. *Correctness of Services and their Composition*. PhD thesis, TU Eindhoven, 2010.

[79] Niels Lohmann, Peter Massuthe, and Karsten Wolf. Operating Guidelines for Finite-State Services. In Jetty Kleijn and Alex Yakovlev, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2007*, volume 4546 of *LNCS*, pages 321–341, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-73093-4. doi: 10.1007/978-3-540-73094-1_20.

[80] Niels Lohmann, H. M. W. Verbeek, and Remco M. Dijkman. Petri Net Transformations for Business Processes — A Survey. In Kurt Jensen and Wil M Aalst, editors, *Transactions on Petri*

Nets and Other Models of Concurrency II, pages 46–63. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-00898-6. doi: http://dx.doi.org/10.1007/978-3-642-00899-3_3.

[81] Alexander Lübbe and Sven Schnägelberger. BPM & O Toolmarktmonitor 2014 - Marktübersicht zu BPM Software für Design & Analyse von Geschäftsprozessen. Technical report, BPM & O, 2014.

[82] Zhilei Ma, Branimir Wetzstein, Darko Anicic, and Stijn Heymans. Semantic Business Process Repository. In *Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007*, 2007.

[83] Harry Maddern, Philip Andrew Smart, Roger S Maull, and Stephen Childe. End-to-end process management: implications for theory and practice. *Production Planning & Control*, 0(0):1–19, 2013. doi: 10.1080/09537287.2013.832821.

[84] Monika Malinova and Jan Mendling. The Effect Of Process Map Design Quality On Process Management Success. *ECIS 2013 Completed Research*, 2013.

[85] Monika Malinova, Henrik Leopold, and Jan Mendling. An Empirical Investigation on the Design of Process Architectures. In *Wirtschaftsinformatik Proceedings*, pages 1197–1211, 2013.

[86] Thomas W Malone, Kevin Crowston, and George A Herman. *Organizing Business Knowledge: The MIT Process Handbook*, volume 22 of *MIT Press Books*. The MIT Press, 2003. ISBN 0262134292. doi: 10.1111/j.0737-6782.2005.116_4.x.

[87] Ronny Mans and N C Russell. Supporting Healthcare Processes with YAWL4Healthcare. *BPM Demos*, 2011. URL http://wwwis.win.tue.nl/$\sim$wvdaalst/publications/p654.pdf.

[88] Ronny S. Mans. *Workflow Support for the Healthcare Domain*. PhD thesis, TU Eindhoven, eindhoven, 2011.

[89] Ronny S. Mans, Nick C. Russel, and Wil M. P. van der Aalst. Inter-workflow support. 2010. URL http://bpmcenter.org/wp-content/uploads/reports/2010/BPM-10-10.pdf.

[90] Axel Martens. On compatibility of web services. *10th. Workshop on Algorithms and Tools for Petri Nets (AWPN 2003)*, 65(12-20):100, 2003.

[91] Axel Martens. Analyzing web service based business processes. *Fundamental Approaches to Software Engineering*, pages 19–33, 2005.

[92] Peter Massuthe. *Operating Guidelines for Services*. PhD thesis, Humboldt University, Berlin, 2009.

[93] Peter Massuthe and K Schmidt. Operating guidelines - an automata-theoretic foundation for the service-oriented architecture. In *Quality Software, 2005. (QSIC 2005). Fifth International Conference on*, pages 452–457, September 2005. doi: 10.1109/QSIC.2005.47.

[94] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. An Operating Guideline Approach to the SOA. *ANNALS OF MATHEMATICS, COMPUTING & TELEINFORMATICS*, 1:35–43, 2005.

[95] Peter Massuthe, Alexander Serebrenik, Natalia Sidorova, and Karsten Wolf. Can I find a partner? Undecidability of partner existence for open nets. *Inf. Process. Lett.*, 108(6):374–378, November 2008. ISSN 0020-0190. doi: 10.1016/j.ipl.2008.07.006.

[96] Joachim Melcher and Detlef Seese. Visualization and Clustering of Business Process Collections Based on Process Metric Values. In *2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 572–575. IEEE, September 2008. ISBN 978-0-7695-3523-4. doi: 10.1109/SYNASC.2008.37.

[97] Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, volume 6 of *LNBIP*. Springer, 2008. ISBN 978-3-540-89223-6.

[98] Jan Mendling. Empirical Studies in Process Model Verification. In Kurt Jensen and Wil M Aalst, editors, *Transactions on Petri Nets and Other Models of Concurrency II*, chapter Empirical, pages 208–224. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-00898-6. doi: 10.1007/978-3-642-00899-3_12.

[99] Jan Mendling, W. M. P. van der Aalst, Boudewijn F Van Dongen, and H. M. W. Verbeek. Errors in the SAP Reference Model. *BPTrends*, 4(6):1–5, 2006.

[100] Jan Mendling, Hajo A Reijers, and Jan Recker. Activity Labeling in Process Modeling: Empirical Insights and Recommendations. *Inf. Syst.*, 35(4):467–482, 2010.

[101] Jan Mendling, Hajo A. Reijers, and W. M. P. van der Aalst. Seven Process Modeling Guidelines (7PMG). *Information and Software Technology*, 52(2):127–136, 2010. ISSN 0950-5849. doi: 10.1016/j.infsof.2009.08.004.

[102] Andreas Meyer and Mathias Weske. Weak Conformance between Process Models and Synchronized Object Life Cycles.

In Xavier Franch, AdityaK. Ghose, GraceA. Lewis, and Sami Bhiri, editors, *Service-Oriented Computing*, volume 8831 of *LNCS*, pages 359–367. Springer Berlin Heidelberg, 2014. ISBN 978-3-662-45390-2. doi: 10.1007/978-3-662-45391-9_25.

[103] Andreas Meyer, Luise Pufahl, Dirk Fahland, and Mathias Weske. Modeling and Enacting Complex Data Dependencies in Business Processes. Technical Report 74, Hasso Plattner Institute at the University of Potsdam, 2013.

[104] Thomas Milde. *Visualization of Business Process Architectures by*. PhD thesis, TU Eindhoven, 2013.

[105] Edgardo Moreira, Christian Fillies, Knowlogy Solutions Ag, and Semtation Gmbh. A Business Process Analysis and Modeling Architecture for E-Government. Technical report, AAAI, 2006.

[106] M Zur Muehlen, DE Wisnosky, and James Kindrick. Primitives: design guidelines and architecture for BPMN models. In *Australasian Conference on Information Systems (ACIS)*, Brisbane, 2010.

[107] Michael Zur Muehlen and Jan Recker. How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering*, CAiSE '08, pages 465–479, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-69533-2. doi: 10.1007/978-3-540-69534-9_35.

[108] Dominic Müller, Manfred Reichert, and Joachim Herbst. Data-driven Modeling and Coordination of Large Process Structures. In *OTM 2007*, volume 4803 of *LNCS*, pages 131–149. Springer, November 2007.

[109] Richard Müller and Andreas Rogge-Solti. BPMN for Healthcare Processes. In Daniel Eichhorn, Agnes Koschmider, and Huayu Zhang, editors, *Proceedings of the 3rd Central-European Workshop on Services and their Composition, ZEUS 2011, Karlsruhe, Germany, February 21–22, 2011*, volume 705 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2011.

[110] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989. ISSN 00189219. doi: 10.1109/5.24143.

[111] A Nigam and N S Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[112] Object Management Group (OMG). Business Process Model and Notation (BPMN) V.2.0. Technical report, OMG, 2011.

[113] Victor Pankratius and Wolffried Stucky. A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*, APCCM '05, pages 79–88, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc. ISBN 1-920-68225-2.

[114] Susanne Patig, Vanessa Casanova-Brito, and Barbara Vögeli. IT Requirements of Business Process Management in Practice – An Empirical Study. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management*, volume 6336 of *LNCS*, pages 13–28. Springer Berlin / Heidelberg, 2010.

[115] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. The Triconnected Abstraction of Process Models. In Umeshwar Dayal, Johann Eder, Jana Koehler, and HajoA. Reijers, editors, *Business Process Management*, volume 5701 of *LNCS*, pages 229–244. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03847-1. doi: 10.1007/978-3-642-03848-8_16.

[116] Artem Polyvyanyy, Sergey Smirnov, and Mathias Weske. On Application of Structural Decomposition for Process Model Abstraction. In *International Conference on Business Process and Services Computing*, pages 110–122, 2009.

[117] Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. Simplified Computation and Generalization of the Refined Process Structure Tree. In Mario Bravetti and Tevfik Bultan, editors, *Web Services and Formal Methods*, volume 6551 of *LNCS*, pages 25–41. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-19588-4. doi: 10.1007/978-3-642-19589-1_2.

[118] Frank Puhlmann and Mathias Weske. Interaction Soundness for Service Orchestrations. In Asit Dan and Winfried Lamersdorf, editors, *Service-Oriented Computing - ICSOC 2006*, volume 4294 of *LNCS*, pages 302–313. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-68147-2. doi: 10.1007/11948148_25.

[119] Corina Raduescu, Hui Min Tan, Malini Jayaganesh, Wasana Bandara, Michael zur Muehlen, and Sonia Lippe. A framework of issues in large process modeling projects. In Jan Ljungberg and Magnus Andersson, editors, *Proceedings of the Fourteenth European Conference on Information Systems, ECIS 2006, Göteborg, Sweden, 2006*, pages 1594–1605, 2006.

[120] Jan Recker. Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, 16(1):181–201, 2010. ISSN 1463-7154. doi: 10.1108/14637151011018001.

[121] Manfred Reichert, Jens Kolb, Ralph Bobrik, and Thomas Bauer. Enabling Personalized Visualization of Large Business Processes through Parameterizable Views. In *27th ACM Symposium On Applied Computing (SAC'12), 9th Enterprise Engineering Track (EE'12)*, pages 1653–1660. ACM Press, March 2012.

[122] H. a. Reijers, J. Mendling, and R. M. Dijkman. Human and automatic modularizations of process models to enhance their comprehension. *Information Systems*, 36:881–897, 2011. ISSN 03064379. doi: 10.1016/j.is.2011.03.003.

[123] Hajo A. Reijers, Ronny S. Mans, and Robert A. van der Toorn. Improved model management with aggregated business process models. *Data & Knowledge Engineering*, 68(2):221–243, 2009. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2008.09.004.

[124] Michael Rosemann. Potential pitfalls of process modeling: part A. *Business Process Management Journal*, 12(2):249–254, March 2006. ISSN 1463-7154. doi: 10.1108/14637150610657567.

[125] Michael Rosemann. Potential pitfalls of process modeling: part B. *Business Process Management Journal*, 12(3):377–384, 2006. ISSN 1463-7154. doi: 10.1108/14637150610668024.

[126] Andreas Rulle and Juliane Siegeris. From a Family of State-Centric PAIS to a Configurable and Parameterized Business Process Architecture. In Shazia Sadiq, Pnina Soffer, and Hagen Völzer, editors, *Business Process Management*, volume 8659 of *LNCS*, pages 333–348. Springer International Publishing, 2014. ISBN 978-3-319-10171-2. doi: 10.1007/978-3-319-10172-9_21.

[127] August-Wilhelm Scheer. *ARIS - Business Process Modeling: Business Process Modeling*. Springer, 3rd edition, 2000. ISBN 978-3-540-65835-1.

[128] August-Wilhelm Scheer, Markus Nüttgens, W. M. P. van der Aalst, Jörg Desel, and Andreas Oberweis. ARIS Architecture and Reference Models for Business Process Management. In Wil Aalst, Jörg Desel, and Andreas Oberweis, editors, *Business Process Management*, volume 1806 of *LNCS*, pages 376–389, Berlin, Heidelberg, March 2000. Springer Berlin Heidelberg. ISBN 978-3-540-67454-2. doi: 10.1007/3-540-45594-9.

[129] August-Wilhelm Scheer, Oliver Thomas, and Otmar Adam. Process Modeling Using Event-Driven Process Chains. In *Process*

Aware Information Systems: Bridging People and Software Through Process Technology., pages 119–146. Wiley Publishing, 2005. ISBN 9780471663065.

[130] Hermann J. Schmelzer and Wolfgang Sesselmann. *Business Process Management in Praxis (Geschäftsprozessmanagement in der Praxis)*. Hanser Fachbuch, 3rd edition, 2003. ISBN 978-3446222984.

[131] Karsten Schmidt. Model-Checking with Coverability Graphs. *Formal Methods in System Design*, 15(3):239–254, 1999. ISSN 0925-9856. doi: 10.1023/A:1008753219837.

[132] Karsten Schmidt. LoLA: A Low Level Analyser. In *ICATPN 2000, International Conference on Theory and Application of Petri nets*, pages 465–474, 2000.

[133] SCOR. *Supply Chain Operations Reference Model (SCOR 11.0)*. APICS Supply Chain Council, Inc., 2012. ISBN 0615202594. URL https://supply-chain.org/scor/11.

[134] Khurram Shahzad, Birger Andersson, Maria Bergholtz, Ananda Edirisuriya, Tharaka Ilayperuma, Prasad Jayaweera, and Paul Johannesson. Elicitation of Requirements for a Business Process Model Repository. In Danilo Ardagna, Massimo Mecella, and Jian Yang, editors, *Business Process Management Workshops*, volume 17 of *LNBIP*, pages 44–55. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00327-1. doi: 10.1007/978-3-642-00328-8_5.

[135] Sergey Smirnov. Structural Aspects of Business Process Diagram Abstraction. In *Commerce and Enterprise Computing, 2009. CEC '09. IEEE Conference on*, pages 375–382, July 2009. doi: 10.1109/CEC.2009.18.

[136] Sergey Smirnov. *Business Process Model Abstraction*. PhD thesis, University of Potsdam, Potsdam, 2011.

[137] Sergey Smirnov, Remco M. Dijkman, Jan Mendling, Mathias Weske, Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson Woo, and Yair Wand. Meronymy-Based Aggregation of Activities in Business Process Models. In Jeffrey Parsons, Motoshi Saeki, Peretz Shoval, Carson Woo, and Yair Wand, editors, *Conceptual Modeling*, volume 6412 of *LNCS*, pages 1–14, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-16372-2. doi: 10.1007/978-3-642-16373-9.

[138] Sergey Smirnov, Hajo A. Reijers, Thijs Nugteren, and Mathias Weske. Business process model abstraction: theory and practice. Technical report, Hasso Plattner Institute, University of Potsdam, 2010.

[139] Sergey Smirnov, Matthias Weidlich, and Jan Mendling. Business Process Model Abstraction Based on Behavioral Profiles. In Paul P Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *Service-Oriented Computing*, volume 6470 of *LNCS*, pages 1–16. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17357-8. doi: 10.1007/978-3-642-17358-5_1.

[140] Sergey Smirnov, Hajo A. Reijers, and Mathias Weske. A semantic approach for business process model abstraction. In *Advanced Information Systems Engineering*, pages 497–511. Springer, 2011.

[141] Sergey Smirnov, HajoA. A. Reijers, Mathias Weske, and Thijs Nugteren. Business process model abstraction: a definition, catalog, and survey. *Distributed and Parallel Databases*, 30(1):63–99, 2012. ISSN 0926-8782. doi: 10.1007/s10619-011-7088-5.

[142] Biplav Srivastava and Debdoot Mukherjee. Organizing Documented Processes. *Services Computing, IEEE International Conference on*, 0:25–32, September 2009. doi: http://doi.ieeecomputersociety.org/10.1109/SCC.2009.32.

[143] Thomas Tullis and William Albert. *Measuring the User Experience. Collecting, Analyzing, and Presenting Usability Metric*. Morgan Kaufmann, 2008.

[144] Moe Tut, David Edmond, Christoph Bussler, Richard Hull, Sheila McIlraith, Maria Orlowska, Barbara Pernici, and Jian Yang. The Use of Patterns in Service Composition. In Christoph Bussler, Richard Hull, Sheila McIlraith, Maria E. Orlowska, Barbara Pernici, and Jian Yang, editors, *WES*, volume 2512 of *LNCS*, pages 28–40, Berlin, Heidelberg, December 2002. Springer Berlin Heidelberg. ISBN 978-3-540-00198-0. doi: 10.1007/3-540-36189-8.

[145] Antti Valmari. The State Explosion Problem. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *LNCS*, pages 429–528, London, UK, UK, 1998. Springer. ISBN 3-540-65306-6.

[146] W. M. P. van der Aalst. Verification of workflow nets. In Pierre Azéma and Gianfranco Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *LNCS*, pages 407–426. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63139-2. doi: 10.1007/3-540-63139-9_48.

[147] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems and Computers*, 08 (01):21–66, 1998. doi: 10.1142/S0218126698000043.

[148] W. M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41 (10):639–650, 1999. ISSN 0950-5849. doi: 10.1016/S0950-5849(99) 00016-6.

[149] W. M. P. van der Aalst and Kees Max van Hee. *Workflow management: models, methods, and systems*. The MIT press, 2004.

[150] W. M. P. van der Aalst and Mathias Weske. The P2P approach to Interorganizational Workflows. In K. Dittrich, A. Geppert, and M. Norrie, editors, *LNCS: Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068, pages 140–156. Springer-Verlag, Berlin, 2001.

[151] W. M. P. van der Aalst, P Barthelmess, C. A. Ellis, and J. Wainer. Workflow modeling using proclets. *Cooperative Information Systems*, pages 198–209, 2000.

[152] W. M. P. van der Aalst, P Barthelmess, C. A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *International Journal of Cooperative Information Systems*, 10(04):443–481, 2001. doi: 10.1142/S0218843001000412.

[153] W. M. P. van der Aalst, A H M ter Hofstede, B Kiepuszewski, and A P Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003. ISSN 0926-8782. doi: 10.1023/A: 1022883727209.

[154] W. M. P. van der Aalst, Ronny Mans, and N C Russell. Workflow Support Using Proclets : Divide , Interact , and Conquer Limitations of Monolithic Workflows. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 1–7, 2009.

[155] W. M. P. van der Aalst, Arjan Mooij, Christian Stahl, Karsten Wolf, Marco Bernardo, Luca Padovani, and Gianluigi Zavattaro. Service Interaction: Patterns, Formalization, and Analysis. *Formal Methods for Web Services*, 5569:42–88, 2009. doi: 10.1007/978-3-642-01918-0.

[156] W. M. P. van der Aalst, K. M. Hee, a. H. M. Hofstede, Natalia Sidorova, H. M. W. Verbeek, M. Voorhoeve, and M. T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing*, 23(3):333–363, August 2010. ISSN 0934-5043. doi: 10.1007/s00165-010-0161-4.

[157] W. M. P. van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. Multiparty Contracts: Agreeing and Implementing Interorganizational Processes. *The Com-*

puter Journal, 53(1):90–106, January 2010. doi: 10.1093/comjnl/bxn064.

[158] Rob J. van Glabbeek and David G. Stork. Query Nets: Interacting Workflow Modules That Ensure Global Termination. In W.M.P. van der Aalst, editor, *Business Process Management*, volume 2678 of *LNCS*, pages 184–199. Springer Berlin / Heidelberg, 2003. ISBN 978-3-540-40318-0.

[159] Jussi Vanhatalo, Hagen Völzer, and Frank Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In BerndJ. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, *Service-Oriented Computing - ICSOC 2007*, volume 4749 of *LNCS*, pages 43–55. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74973-8. doi: 10.1007/978-3-540-74974-5_4.

[160] Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. *Data & Knowledge EngineeringKnowledge Engineering*, 68(9):793–818, 2009. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2009.02.015.

[161] H. M. W. Verbeek, T. Basten, and W M P van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001. ISSN 0010-4620, 1460-2067. doi: 10.1093/comjnl/44.4.246.

[162] Matthias Weidlich, Artem Polyvyanyy, Nirmit Desai, and Jan Mendling. Process compliance measurement based on behavioural profiles. In *Proceedings of the 22nd international conference on Advanced information systems engineering*, CAiSE'10, pages 499–514, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13093-3, 978-3-642-13093-9.

[163] Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Trans. Software Eng.*, 37(3):410–429, 2011.

[164] Daniela Weinberg. Efficient Controllability Analysis of Open Nets. In Roberto Bruni and Karsten Wolf, editors, *Web Services and Formal Methods*, volume 5387 of *LNCS*, pages 224–239. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01363-8. doi: 10.1007/978-3-642-01364-5_14.

[165] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2nd edition, 2012. ISBN 3540735216.

[166] Claes Wohlin, Per Runeson, Martin Hst, Magnus C Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software*

Engineering. Springer Berlin Heidelberg, 2012. ISBN 3642290434, 9783642290435.

[167] Karsten Wolf. Generating Petri net state spaces. *Proceedings Petri Nets '07*, pages 29–42, 2007. ISSN 03029743.

[168] Wolfang Reisig. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien*. Leitfäden der Informatik. Vieweg+Teubner, 1 edition, 2010.

[169] Zhiqiang Yan and Paul Grefen. A Framework for Business Process Model Repositories. In Michael zur Muehlen and Jianwen Su, editors, *Business Process Management Workshops*, volume 66 of *LNBIP*, pages 559–570. Springer, 2011. ISBN 978-3-642-20510-1. doi: 10.1007/978-3-642-20511-8_51.

[170] Zhiqiang Yan, Remco M. Dijkman, and Paul Grefen. Business Process Model Repositories - Framework and Survey. `cms.ieis.tue.nl/Beta/Files/WorkingPapers/Beta_wp292.pdf`, 2009. URL `cms.ieis.tue.nl/Beta/Files/WorkingPapers/Beta_wp292.pdf`.

[171] Zhiqiang Yan, Remco M. Dijkman, and Paul Grefen. Business process model repositories - Framework and survey. *Information and Software Technology*, 54(4):380–395, April 2012. ISSN 09505849. doi: 10.1016/j.infsof.2011.11.005. URL `http://www.sciencedirect.com/science/article/pii/S0950584911002291`.

[172] Robert K. Yin. *Case study research: Design and methods*, volume 5. Sage Publications, Inc., 5th. edition, 2014.

[173] Sira Yongchareon, Chengfei Liu, and Xiaohui Zhao. A Framework for Behavior-Consistent Specialization of Artifact-Centric Business Processes. In *BPM*, pages 285–301. Springer, 2012.

[174] John A Zachman. A Framework for Information Systems Architecture. *IBM Syst. J.*, 26(3):276–292, September 1987. ISSN 0018-8670. doi: 10.1147/sj.263.0276.

[175] Xiaohui Zhao, Chengfei Liu, Yun Yang, and Wasim Sadiq. CorPN: managing instance correspondence in collaborative business processes. *Distributed and Parallel Databases*, 29 (4):309–332, March 2011. ISSN 0926-8782. doi: 10.1007/s10619-011-7080-0.