Universität Stuttgart

Resource-driven Processes

Concept, Use, and Incorporation

Von der Graduiertenschule GSaME Graduate School of Excellence advanced Manufacturing Engineering der Universität Stuttgart zur Erlangung der Würde eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von

Celal Timurhan Sungur

aus Ankara (Türkei)

Hauptberichter:	Prof. Dr. Dr. h.c. Frank Leymann
Mitberichter:	Univ. Prof. Dr. Schahram Dustdar
Tag der mündlichen Prüfung:	09. Oktober 2018

Institut für Architektur von Anwendungssystemen
der Universität Stuttgart

2018

CONTENTS

ZUSAMMENFASSUNG

Die Erreichung der Ziele von Organisationen erfordert die Ausführung gewisser Geschäftsprozesse (kurz Prozesse). Die Modellierung, die Nutzung, und die Verbesserung des vorhandenen Wissens über diese Prozesse ermöglichen die Einrichtung organisatorischer Erfolgsmethoden. Eine solche Einrichtung wird typischerweise durch die handlungsorientierte Modellierung, Nutzung und Verbesserung der Prozesse in Bezug auf ihre Aktivitäten und deren Reihenfolge durchgesetzt. Zudem können diese modellierten Aktivitäten mittels IT-Infrastrukturen automatisiert und koordiniert werden. Jedoch sind Aktivitäten und deren Reihenfolge nicht immer (i) vorhersehbar während der Modellierung und (ii) bei jeder Prozessausführung wiederholt. Diese Abweichungen von Aktivitäten und deren Reihenfolge in unterschiedlichen Ausführungen eines Prozesses beeinträchtigt die Verwendbarkeit der handlungsorientierten Modellierungsansätze und weckt einen Bedarf an einem Ansatz (i) zur Unterstützung menschlicher Akteure von derartigen Prozessen und (ii) zur Reproduzierbarkeit deren gewünschten Ergebnisse. Außerdem erhöht die zunehmende Nachfrage nach individualisierten Produkten und Lösungen diesen Bedarf weiter, da jedes solche Produkt und jede solche Lösung maßgeschneiderte Aktivitäten mit verschiedenen Reihenfolgen fordern.

In dieser Dissertation wird ein ressourcengetriebener Ansatz zur Modellierung und Ausführung der Prozesse vorgestellt. Dieser Ansatz ermöglicht

(i) die Unterstützung menschlicher Akteuren und (i) die Reproduzierbarkeit der gewünschten Ergebnisse von Prozessen mittels der automatisch bereitgestellten zusammenhängenden Ressourcen. Um Definitionen ressourcengetriebener Prozesse zu erstellen, wird eine formale ressourcengetriebene Modellierungssprache von Prozessen mit unterschiedlichen Modellierungselementen vorgestellt, so wie Ziele, Fähigkeiten und zusammenhängende Ressourcen. Zur Evaluation und Validierung dieses Ansatzes wurden Umfragen mit 416 Personen durchgeführt. Die Ergebnisse der Umfragen untermauern unsere Behauptung, dass (i) die Unterstützung menschlicher Akteure von Prozessen und (ii) die Reproduzierbarkeit der Ergebnisse durch unseren Ansatz ermöglicht werden können.

Um ressourcengetriebene Prozesse in Organisationen systematisch nutzen zu können, stellen wir einen Lebenszyklus mit vier Phasen vor. Die erste Phase enthält Schritte zur Vorbereitung einer IT-Infrastruktur, um die restlichen Phasen zu ermöglichen. Bei der zweiten Phase startet die Modellierung ressourcengetriebener Prozesse mit Zielen und endet mit der Modellierung zusammenhängender Ressourcen. Hierauf erfolgt die Ausführung von modellierten ressourcengetriebenen Prozessen in der Phase 3. Nach der Initialisierung der ressourcengetriebenen Prozesse werden ausgewählte zusammenhängende Ressourcen automatisch bereitgestellt, damit sie gemeinschaftlich Ziele des Prozesses verwirklichen. Daraufhin werden basierend auf den bei Ausführungen entstandenen Ressourceninteraktionen in der Phase 4 ressourcenzentrische Empfehlungen erzeugt, die Geschäftsexperten bei der Modellierung von ressourcengetriebenen Prozessen leiten. Die im Lebenszyklus eingeführten Konzepte wurden durch prototypische Implementierungen und eine Befragung validiert und bzw. evaluiert.

Nicht zuletzt wird die Aufnahme ressourcengetriebener Prozesse in handlungsorientierte Prozesse untersucht. Somit wird die neue Art von einer Aktivität die „kontextsensitive Aktivität“, die die Ausführung an der vorhanden Situation anpasst, vorgestellt. Kontextsensitive Aktivitäten wurden dadurch validiert, dass ein Werkzeug von handlungsorientierten Prozessen erweitert wurde, um diese Aktivitäten zu unterstützen.

Abstract

Reaching organizational goals requires executing business processes. Modeling, using, and improving existing knowledge about business processes establishes organizational best practices. A common method for this is accomplished in an activity-oriented way by modeling, using, and improving business processes based on recurring activities and their order. Furthermore, modeled activities and their coordination can be automated with the help of IT infrastructures to increase automation and support for actors. Unfortunately, activities and their order in business processes are not always (i) foreseeable at modeling time or (ii) repeated in different executions. This variation of activities and their order among business processes decreases the usefulness of activity-oriented modeling approaches and raises the need for another approach to (i) support such business processes and (ii) reproduce desired outcomes. In addition, this need is intensified by increasing demands toward individualized products and solutions, as each product and solution can require custom-tailored activities in a different order.

In this work, we introduce a resource-driven approach for modeling and executing business processes. Our approach relies on automatically allocated interrelated resources for supporting actors participating in business processes and reproducing their desired outcomes. To create definitions of business processes in a resource-driven way, we present a formal resource-

driven process modeling language capable of specifying business processes in terms of their goals, capabilities, and interrelated resources. To evaluate and validate our approach, we conducted a survey with 416 participants. Results of the survey confirm our claims regarding (i) increased support for actors of business processes and (ii) the reproducibility of their desired outcomes using our resource-driven approach.

For using resource-driven processes in organizations, we present a resource-driven process management life cycle involving four phases. The first phase of the life cycle describes steps needed for preparing an IT infrastructure enabling the steps conducted in other phases of the life cycle. In the second phase, business experts model resource-driven processes by starting with specifying goals and ending with selecting appropriate interrelated resources. The execution of resource-driven processes takes place in the third phase. Upon initializing modeled resource-driven processes, interrelated resources of resource-driven processes are automatically allocated, if applicable. The allocated resources collaboratively work toward the goals specified in definitions of resource-driven processes resulting in interactions between resources. In the fourth phase, these interactions are analyzed to generate resource-centric recommendations to guide business experts during modeling. We implemented a series of prototypes and conducted an expert survey to validate and evaluate the life cycle.

Finally, we present the means of incorporating resource-driven processes into activity-oriented business process models. Therefore, we present a new type of activity construct called context-sensitive activity, adapting the execution based on the current situation. We validated the concept of context-sensitive activities by extending a tool for activity-oriented business processes to support context-sensitive activities.

CHAPTER 1

INTRODUCTION

Organizations set and achieve organizational goals, such as fixing a defect in a mobile device. Accomplishing these goals requires enacting business processes containing a set of activities in a specific order [Wes10; LR00; DVT05]. When business processes are executed repeatedly, these can be documented based on the reusable information shared between different enactments aimed at the same goals. For example, organizations can model activities and their order (i.e., business logic) to reach a certain goal resulting in activity-oriented[1] modeling of business processes.

Activity-oriented business process models facilitate various benefits for organizations, such as the automation and coordination of activities. Therefore, organizations can create web services implementing activities contained in business process models and coordinate these using another web service orchestrating their execution. Unfortunately, creating traditional activity-oriented process models is an expensive task [LR00]; thus, it is not always feasible to model business processes in an activity-oriented fashion. Furthermore, certain business processes involve ad hoc activities, making these processes difficult or impossible to model based on their activities before

[1]Nurcan [Nur08] uses the term activity-oriented

their enactments. For example, activities conducted in a business process to fix a defect of a mobile device are not predictable before the enactment, as actors conduct these in an ad hoc fashion based on the definition of the defect. During these business processes, human actors, or shortly actors, solve emerging problems based on their expertise, education, and experience [Dav05]. Thus, activities and their order in such business processes (i) are created on the fly during their execution and (ii) typically are not repeated during different enactments of the same process [DMR15].

The increasing demand toward individualized products and solutions will result in business processes that more frequently require individualized solutions [Wes13; EL16; LFK+14; BFKR14]. Interestingly, creating these individualized products and solutions requires carrying out similarly individualized activities in a certain order. For example, a customer of a mobile device producer can require an individualized mobile device with water resistance, whereas another customer can desire other functionalities, such as a longer battery life. Business processes aimed at both these individualizations will have the same goal although the activities and their order can be radically different because of changing customer requirements. Thus, modeling and executing business processes representing these in an activity-oriented fashion becomes unsuitable [Dav10; GOR11].

Other approaches enabling the modeling of business processes introduce modeling of other reusable elements, such as case management [Obj16] and activity-centered computing [MCF05; MGM+06; BKHM07]. For example, using the case management approach [Obj16], actors can store files, folders, and XML files relevant for the business process. Furthermore, using activity-centered computing [MCF05; MGM+06; BKHM07], it is possible to document IT tools required for completing business processes. However, case management and activity-centered computing are not designed to represent every category of resources. To this end, a resource is "a stock or supply of money, materials, staff, and other assets that can be drawn on by a person or organization in order to function effectively" [Oxf17]. Consequently, representing resources can be important for the effectiveness of business processes and these approaches fail to represent the actual business process execution in terms of their resources. For example, it is not possible to represent a simulation machine required for executing simulations because of the limited availability of different types of resources, although a simulation machine can be important for a business process aimed at fixing a defect in a mobile device. Moreover, actors use other resources with different access rights during the execution of business processes, resulting in different relationships between different resources. Thus, missing interrelations between resources in case management and activity-centered computing approaches limit the definition of interrelated resources, such as a team member managing a Wiki service, and the other members using the service. Because of limiting the representation of resources and their relationships, actors can waste time with unproductive activities, such as searching for certain resources, allocating these, and configuring them for the appropriate access rights. These unproductive activities can decrease the performance of business processes involving actors.

Considering the increasing trends toward business processes aimed at creating individualized services and individualized products, such a decrease in performance cannot be tolerated. As a result, there is an increasing need for (i) supporting actors of business processes and (ii) reproducing their desired outcomes without basing these on the reusable activities and their sequence. To this end, supporting actors of business processes is aimed at (i) increasing the number and performance of productive activities of actors and (ii) decreasing their avoidable unproductive activities. For example, actors should be able to start directly debugging a mobile device to investigate a defect instead of spending time with setting up the respective development environment. Furthermore, reproducing desired outcomes of business processes requires documenting business processes with sufficient information to produce their desired outcomes repeatedly. To address this, we introduce a resource-driven process modeling and execution approach relying on automatically allocated interrelated resources to (i) support actors in business processes and (ii) reproduce their desired outcomes.

Moreover, we present a management life cycle involving four phases for enabling, defining, executing, and improving resource-driven processes in organizations. In various cases, resource-driven processes are initialized by a business process defined in an activity-oriented way, such as a manual maintenance process initialized in a production process of a mobile device. This requires a mechanism for incorporating resource-driven processes into activity-oriented processes. Therefore, we present a new type of activity capable of invoking resource-driven processes in business processes defined in an activity-oriented way.

In the following section, we discuss different types of business processes in organizations to increase comprehensibility of the goals of this thesis. We present our research questions and contributions in Section 1.2. Finally, we explain the contents of this work in Section 1.3.

1.1 Informal Processes

Understanding the rest of this work requires a solid terminology to avoid confusion. Therefore, we clarify the basic terms used, such as informal processes, in the rest of this work. To this end, informal processes are a categorization of different types of business processes, and resource-driven processes are an approach to modeling and executing business processes, which is detailed in Chapter 3. As our main contributions are centered on an approach capable of providing support for actors of business processes involving unpredictable activities, we first investigate different types of organizational business processes. To categorize different types of business processes in organizations, we use the business process spectrum of Di Ciccio et al. [DMR15] due to its focus on the variable predictability of the business logic of business processes and reasons for this variability.

The business process spectrum of Di Ciccio et al. [DMR15] presents five main types of business processes: (i) structured business processes, (ii) structured business processes with ad hoc exceptions, (iii) unstructured business processes with predefined process fragments, (iv) loosely structured business processes, and (v) unstructured business processes. Structured business processes contain a fixed set of activities with the same order between different enactments of business processes with the same goals, such as production processes and administrative processes [LR00; DMR15]. As activities and their order do not change during different enactments of the same business process, it is possible to predict the activities and their order beforehand. Consequently, organizations can model interrelated activities in an activity-oriented fashion resulting in business process models using different business process modeling languages, such as Business Process Model and Notation (BPMN) and WS-Business Process Execution Language (BPEL), and Yet Another Workflow Language (YAWL) [Obj11; OAS07b; vdAtH05]. Furthermore, organizations can use business process models to automate enactments of their business processes with the help of their IT infrastructures [LR00; Wes10]. Therefore, technical experts create executable business process models, such as executable BPMN models [Sil11], and configure the necessary IT infrastructure for automating business processes. For example, technical experts develop web services automating activities and refer to these web services in activity-oriented business process models. After creating executable business process models, organizations deploy these business process models on their supporting business process execution engines capable of coordinating the execution of different activities with the help of the configured IT infrastructure. Upon the initialization of business process models, business process execution engines create instances of the models also known as business process instances.

During the execution of business processes, exceptional situations arise, such as the breakdown of manufacturing machines in a production process, resulting in the execution of a repair process. Structured business processes capable of handling such exceptional situations are structured business processes with ad hoc exceptions. When exceptions are predictable at modeling time, organizations can cope with these by adding the necessary handling logic into business process models. Moreover, organizations can create the necessary handling logic at runtime in an ad hoc fashion to deal with unpredictable exceptional situations [SK10]. Consequently, organizations can create the business logic for handling exceptions both at modeling time and runtime of structured business processes with ad hoc exceptions.

Unstructured business processes with predefined process fragments are business processes that are assembled at runtime based on the problem context, organizational rules, policies, and regulations. These organizational rules, policies, and regulations resolve into predefined interconnected groups of process elements, such as activities, activity placeholders, and dependencies that are process fragments [SLM+10]. When process fragments can be automatically executed, the assembly process can be automated by defining certain rules for selecting process fragments [Ebe14]. For example, organizations can use adaptive pervasive flows [BLMP09] that are business processes capable of adapting themselves to their actual environments. Moreover, this assembly can be manually performed by actors during the enactment based on their expertise, education, and experience, depending on the considered business process [Dav05]. An example of such a business process is an insurance claim [DMR15], as actors of an insurance claim construct the actual business logic during enactment depending on the available insurance policies, regulations, and information regarding the current case.

When organizational rules, policies, and regulations do not resolve into a set of predefined process fragments but, rather, resolve into a limited set of activities, business processes become loosely structured business processes. For example, treating a patient based on his or her symptoms can be considered in this category of business processes, as the possible set of activities is limited [DMR15]. In this case, the structure of the assembled processes is based on a limited set of activities during runtime.

Relaxing the condition from being limited to being unlimited on the set of possible activities results in unstructured business processes. More specifically, activities of unstructured business processes are not predictable, as the activities need change based on different factors, such as the experience of actors, changing requirements, changing goals, and the current situation. Furthermore, the freedom of actors in terms of possible activities promotes explorations and facilitates innovative solutions [Wal14; Mar91]. For example, a business process aimed at investigating and fixing a defect in a mobile product can be partially considered in this category, as it requires the enactment of unpredictable activities in an unanticipated order depending

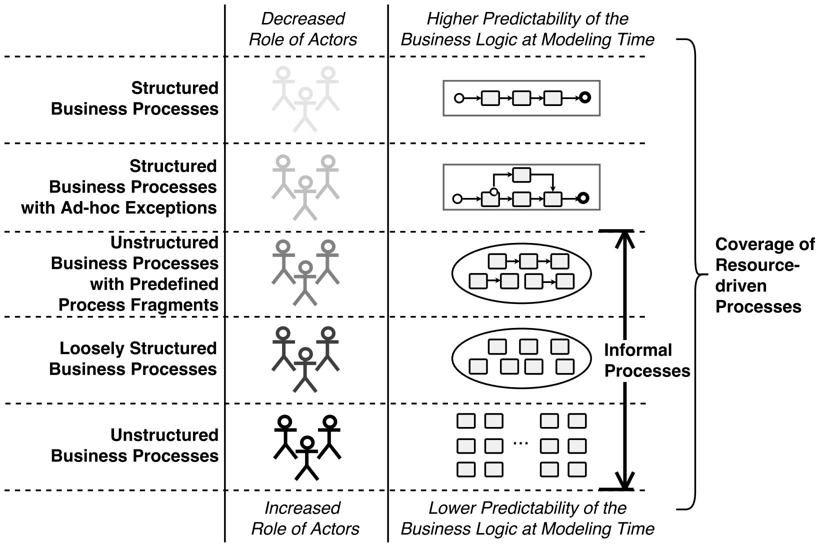

Decreased Role of Actors
Higher Predictability of the Business Logic at Modeling Time
Structured Business Processes
Structured Business Processes with Ad-hoc Exceptions
Unstructured Business Processes with Predefined Process Fragments
Loosely Structured Business Processes
Unstructured Business Processes
Informal Processes
Coverage of Resource-driven Processes
Increased Role of Actors
Lower Predictability of the Business Logic at Modeling Time

Figure 1.1: Positioning of informal processes in the business process spectrum of Di Ciccio et al. [DMR15].

on the nature of the failure. Unstructured processes are highly collaborative in their nature and involve activities creating, applying, and disseminating the knowledge, which is knowledge-intensive [Dav10].

Figure 1.1 presents a summary of the explained business spectrum based on the business spectrum of Di Ciccio et al. [DMR15]. In structured business processes, actors give up their responsibility for coordinating different activities to business process execution engines. Business process execution engines coordinate the execution of activities as prescribed in business process models. Consequently, the roles of actors are restricted to the activities assigned to them by business process execution engines. Conversely, actors of structured business processes with ad hoc exceptions have more responsibilities than within structured business processes, as they additionally have the role of reacting to unpredictable ad hoc exceptions during business processes. This role increases in the case of unstructured business processes with predefined fragments, as actors can be responsible for assembling the resulting business logic using available process fragments manually (e.g., assembling cannot be done automatically because of high implementation costs). Available process fragments of unstructured business processes with predefined fragments contain interrelated activities, whereas loosely structured business processes relax this requirement by containing a limited set of possible independent activities. Therefore, actors select and assemble activities in the right order based on their expertise, education, and experience [Dav05]. This additional ordering of activities, instead of ordering process fragments containing ordered activities, increases the flexibility of actors but, similarly, increases their role in the resulting business logic. Finally, unstructured business processes leave the whole responsibility for completing a business process to their actors, resulting in an even more important role for them.

The increasing role of actors is inversely proportional to the predictability of the business logic at modeling time of business processes, as shown in Figure 1.1. Moreover, the trend of the decreasing predictability of business logic toward unstructured business processes is a consequence of the increasing number of unanticipated situations that can happen during business processes. Increasing the number of unanticipated situations typically leaves management of these situations to the actors. Moreover, documenting all execution paths of business processes with a lower predictability of the business logic in an activity-oriented method becomes unfeasible, such as a business process aimed at fixing a defect of a product [GOR11; Dav10]. Because of this unfeasibility, organizations enact such business processes in an informal way without creating more costly activity-oriented business process models, resulting in informal processes [MGM+06]. More precisely, we use the following definition to specify informal processes:

Definition 1 (Informal Processes). Informal processes are human-centric business processes with a low predictability of their business logic at modeling time, making activity-oriented documentation of these processes unsuitable. Furthermore, the reason of their human-centric nature is the dependency of the resulting business logic during process enactments on the decisions of the actors. ♣

Figure 1.1 represents the position of the informal processes in the business process spectrum. To this end, structured business processes with ad hoc exceptions are not considered informal processes due to their foreseeable structure at modeling time. In contrast, the resulting business logic of unstructured business processes with predefined process fragments, loosely structured business processes, and unstructured business processes may completely depend on the actors. Thus, we classify these as informal processes. Next, we describe our research questions and contributions in detail.

1.2 Research Questions and Contributions

Our contributions can be considered in two major blocks, as illustrated in Figure 1.2. The first block, contributions 1, 2, 3, and 4, comprises the concept of resource-driven processes and their management in organizations. More specifically, the first contribution of the first block is a resource-driven approach for modeling and executing business processes. For managing resource-driven processes in organizations, we proposed a Resource-driven Process Management (RPM) life cycle involving different phases, such as modeling, executing, and discovering resource-driven processes. The rest of the contributions residing on top of the first contribution in Figure 1.2 are part of this RPM life cycle. Therefore, these contributions are presented together in Section 1.2.2. The second block is the fifth contribution, representing the concept of a new type of activity called the context-sensitive activity in activity-oriented business process models capable of incorporating resource-driven processes into activity-oriented business process models. We consider this a separate block of contributions, as the context-sensitive activities are a standalone concept and enable the incorporation of business processes designed using different approaches, such as activity-oriented business process modeling. In the following, we start with the fundamental contribution of our work, the resource-driven processes.

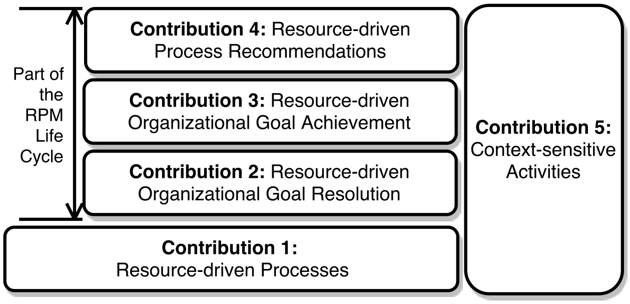

Part of the RPM Life Cycle
Contribution 4: Resource-driven Process Recommendations
Contribution 3: Resource-driven Organizational Goal Achievement
Contribution 2: Resource-driven Organizational Goal Resolution
Contribution 1: Resource-driven Processes
Contribution 5: Context-sensitive Activities

Figure 1.2: Contributions of this thesis.

1.2.1 Resource-driven Processes

The increasing demand toward individualized products and services increases the need for executing informal processes in organizations that provide individualized solutions [EL16; LFK+14; BFKR14]. Moreover, many informal processes, such as (i) innovation and (ii) crisis management processes, set the degree of the competitiveness of organizations by (i) creating disruptive innovations, increasing the competitiveness of an organization and (ii) planning the management of crises to stay competitive [Dav05; SH03; Bur04]. Consequently, increasing the performance of informal processes is in the interest of organizations. To increase their performance, organizations can provide support for actors leading informal processes to successful outcomes. Furthermore, organizations will typically desire to reproduce successful outcomes of informal processes repeatedly, if these are enacted repeatedly.

The concept of informal processes also covers unstructured business processes with changing activities from one enactment to another. Thus, documenting interrelated activities and executing informal processes based on the documentation to support actors and reproduce desired outcomes of informal processes is not an option, as the documented activities can change from execution to execution. For example, activities and their order for a business process aimed at fixing a defect of a mobile device can radically change from one enactment to another due to a change in the considered defect. Consequently, there is a need for an approach capable of supporting actors of informal processes and reproducing their desired outcomes, even if no reusable (interrelated) activities between two different executions of an informal process exist. To address this need, we present the resource-driven processes as our initial contribution in this work.

Contribution 1 (Resource-driven Processes). Resources are the most fundamental requirements for achieving goals in organizations. We rely on this fact and introduce a novel approach, defining business processes based on their interrelated resources. Definitions of resource-driven processes specify the means of achieving goals of business processes in terms of required capabilities and interrelated resources, providing these capabilities. Upon the initialization of these definitions, interrelated resources required are (automatically) allocated for accomplishing goals of business processes. Consequently, allocated resources support actors of business processes during their executions. Furthermore, allocated resources produce desired outcomes of business processes autonomously. In our contribution, we explain the concept of resource-driven processes and present a formal language for defining resource-driven processes.

1.2.2 Using Resource-driven Processes in Organizations

The introduction of resource-driven processes raises a question regarding their use in organizations. More specifically, it should be detailed how resource-driven approaches should be implemented in organizations. Therefore, we introduce the Resource-driven Process Management life cycle involving further contributions of this work. The Resource-driven Process Management life cycle enables systematically modeling, executing, and improving resource-driven processes.

Naturally, managing resource-driven processes requires existing definitions of resource-driven processes. Furthermore, the systematic creation of definitions of resource-driven processes is an open research question, because the modeling of resource-driven processes has not been investigated so far. Therefore, in our next contribution, we present concepts and detailed steps required for creating definitions of resource-driven processes using the language presented in our first contribution.

Contribution 2 (Resource-driven Organizational Goal Resolution). Business processes of organizations aim at achieving organizational goals. Consequently, goals of business processes are known before executing the business processes. This fact presents a good starting point for creating definitions of resource-driven processes. In our modeling method, business experts first specify the goals of the resource-driven processes. These are detailed with capabilities that are provided by organizational resources. Finally, business experts associate interrelated resources required for accomplishing these goals. Thus, definitions of resource-driven processes present a resolution of organizational goals into interrelated resources. In our contribution, we detail concepts and modeling steps involved during the resolution of organizational goals into interrelated resources.

Definitions of resource-driven processes contain required interrelated resources to achieve business process goals. These interrelated resources include human, IT, information, and physical resources, such as developers, Wiki services, integrated development environments, and mobile devices. Leaving the allocation of these resources specified in definitions of resource-driven processes to actors results in a waste of time containing unproductive activities to set up their workspace for conducting productive activities, such as setting up a development environment.

The automated allocation of interrelated resources upon the initialization of resource-driven processes can avoid such unproductive activities and increase the focus of actors on the actual goals of business processes. Furthermore, such an automated allocation of interrelated resources will increase the support provided for actors of business processes by increasing their productivity, such as by making the needed information automatically ready and accessible upon initialization. However, the automated allocation of interrelated resources is not trivial. It requires new concepts capable of managing different types of resources in a unified manner, such as human resources and IT resources. Therefore, our next contribution is concepts and detailed steps involved for enacting resource-driven processes.

> **Contribution 3 (Resource-driven Organizational Goal Achievement).** Definitions of resource-driven processes specify organizational goals and interrelated resources required for reaching these goals. To achieve the goals, the required interrelated resources are automatically allocated for the goals of business processes upon the initialization of definitions of resource-driven processes. During business processes, actors autonomously work toward business processes' goals to accomplish these. Furthermore, allocated resources of business processes are deallocated automatically upon the finalization of the business processes. In our contribution, we present steps and their related concepts for enacting resource-driven processes to enable goal achievement in a resource-driven way by initializing definitions of resource-driven processes.

Initializing definitions of resource-driven processes results in the allocation of interrelated resources providing required capabilities. Upon initialization, the allocated resources collaboratively work toward specified goals of resource-driven processes. Furthermore, during resource-driven processes, ad hoc collaborations can take place due to newly emerging requirements, resulting in the participation of resources missing in definitions of the resource-driven processes. For example, a developer can consult with another organization member in an ad hoc fashion without updating the respective definition of the resource-driven process.

Consequently, definitions of resource-driven processes can diverge from the actual executions. This divergence decreases the support provided by resource definitions due to the created gap between the definitions and actual executions. To avoid this decrease, we present our next contribution, analyzing existing enactments of resource-driven processes to present resource-centric recommendations at modeling time to business experts.

Contribution 4 (Resource-driven Process Recommendations).
Resources allocated for business processes collaboratively work toward the goals of the business processes. A natural consequence of these collaborations are interactions, which are stored in different event logs in locations, such as Git repositories. Furthermore, analyzing interactions can provide insight regarding the contributions and degree of contributions of different resources in business processes. Therefore, we present concepts capable of analyzing these interactions to generate resource-centric recommendations of resources, capabilities provided by the resources, and definitions of resource-driven processes containing resource recommendations. Thus, this contribution presents a means of providing recommendations for business experts adapting definitions of resource-driven processes based on interactions created during resource-driven processes.

Definitions of resource-driven processes can be initialized during the execution of business processes modeled in an activity-oriented fashion. In the following section, we present our contribution for incorporating definitions of resource-driven processes into activity-oriented business process models.

1.2.3 Incorporating Resource-driven Processes into Activity-oriented Business Process Models

Executing business processes requires carrying out a set of interrelated activities. When interrelated activities repeat during different executions of a business process aimed at the same organizational goals, these activities and their interrelations can be captured in activity-oriented business process models. Furthermore, activities documented for a business process can be recursively described in terms of other business processes, which are sub-processes [LR00]. These sub-processes can be informal processes. For example, an organization producing mobile devices can predict a failure in their simulation machines and can initiate the execution of a maintenance process without stopping the simulations completely. Obviously, such a maintenance process can be expressed using the resource-driven process modeling approach introduced in this work. Thus, there is a need for a means of invoking resource-driven processes from activity-oriented business process models. In our next contribution, we present a novel activity construct enabling the incorporation of resource-driven processes into activity-oriented business process models.

Contribution 5 (Context-sensitive Activities). The decision to initialize a resource-driven process from a business process will typically depend on the current context of the business process. Moreover, based on the current situation, it can be possible to optimize further a resource-driven process, such as reducing the number of resources in a resource-driven process based on the current situation. To address these aspects, we introduce a new type of activity for activity-oriented business process modeling called context-sensitive activities. During the execution of context-sensitive activities, multiple business processes are activated in parallel, depending on the current context and goals of the business processes. Furthermore, each activated business process is optimized using its custom optimization logic, if available. Thus, this contribution provides a generic method of incorporating all kinds of business processes in a context-sensitive and a goal-oriented fashion independently from the definition languages into activity-oriented business process models, such as resource-driven processes and activity-oriented processes.

1.3 Outline of the Thesis

In the following, we describe the outline of this thesis and the associated contributions.

- Chapter 2 – *Fundamentals and Related Work:* This chapter explains the underpinnings of our work.
- Chapter 3 – *Resource-driven Processes:* This chapter explains the concept of resource-driven processes in detail, including a formal definition language called Resource-driven Process Modeling Language. Relevant contributions: Contribution 1.

- Chapter 4 – *Using Resource-driven Processes in Organizations:* This chapter explains the details of using resource-driven processes in organizations. Therefore, the Resource-driven Process Management life cycle is presented. Relevant contributions: Contribution 2, Contribution 3, and Contribution 4.
- Chapter 5 – *Incorporating Resource-driven Processes into Activity-oriented Business Processes:* This chapter explains the details of Context-sensitive Activities, that is, a new type of activity in activity-oriented business process models for incorporating business processes defined in a resource-driven way. Relevant contributions: Contribution 5.
- Chapter 6 – *Evaluation and Validation of Concepts:* This chapter presents a survey that we conducted to evaluate resource-driven processes described in Chapter 3. Furthermore, we explain our prototypes implementing the Resource-driven Process Management life cycle explained in Chapter 4 and the Context-sensitive Activities described in Chapter 5. Relevant contributions: Contribution 1, Contribution 2, Contribution 3, Contribution 4, and Contribution 5.
- Chapter 7 – *Conclusion and Outlook:* This chapter presents the conclusions of this work and gives a set of future directions to be followed.

1.4 List of Peer-reviewed Publications

The following presents the list of peer-reviewed publications done in the scope of this work, such as conference proceedings and workshops, in chronological order starting from the oldest.

- C. T. Sungur, O. Kopp, and F. Leymann. "Supporting Informal Processes." In: *The 6th Central European Workshop on Services and their Composition (ZEUS 2014)*. Vol. 1140. CEUR-WS, Feb. 2014, pp. 49–56
- C. T. Sungur et al. "Informal Process Essentials." In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014)*. IEEE, Sept. 2014, pp. 200–209

- C. T. Sungur et al. “Transforming Collaboration Structures into Deployable Informal Processes.” In: *Engineering the Web in the Big Data Era: ICWE 2015*. Springer, June 2015, pp. 231–250
- C. T. Sungur et al. “Executing Informal Processes.” In: *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS ’15)*. ACM, Dec. 2015, 54:1–54:10
- C. T. Sungur et al. “Context-sensitive Adaptive Production Processes.” In: *48th CIRP Conference on Manufacturing Systems (CIRP CMS 2015)*. Vol. 41. Elsevier, June 2015, pp. 147–152
- C. T. Sungur et al. “Identifying Relevant Resources and Relevant Capabilities of Collaborations – A Case Study.” In: *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW 2016)*. IEEE, Nov. 2016, pp. 1–4
- C. T. Sungur et al. “Identifying Relevant Resources and Relevant Capabilities of Informal Processes.” In: *19th International Conference on Enterprise Information Systems (ICEIS 2017)*. SciTePress, Apr. 2017, pp. 295–307

CHAPTER 2

Fundamentals and Related Work

In this chapter, we present fundamentals and related work for understanding this work. We first present a set of properties of informal processes important in Section 2.1. These properties create a basis for a set of requirements for supporting human actors of informal processes and for reproducing their desired outcomes, which we present in the next chapter in Section 3.1. Hereafter, we present an overview of managing business processes of organizations in Section 2.2. During the management of business processes, organizations use different approaches for modeling and executing these processes, such as the activity-oriented modeling and execution. Such modeling and execution approaches are highly relevant, as we present a business process modeling and execution approach in this work. Therefore, we explain these in Section 2.3. Finally, we present the computing model cloud computing used during the implementation of the presented concepts in Section 2.4. Please note that this chapter does not present a comparison between concepts introduced in this work and existing concepts from the literature.

Such a comparison will be presented at the end of respective chapters after explaining different concepts.

2.1 Properties of Informal Processes

The following properties present a set of characteristics of informal processes that (i) decrease the applicability of activity-oriented approaches for such processes and (ii) must be considered during modeling and enacting such processes. Further properties can be found in [DMR15; Dav05].

2.1.1 Unpredictable and Unrepeatable Process Enactments

Informal processes typically involve activities requiring actors to apply their expertise, education, and experience to accomplish goals of the processes collaboratively [Dav05]. Consequently, the business logic created for an informal process depends on and can vary based on expertise, education, and experience. Furthermore, actors create the business logic for achieving certain goals of informal processes. Although a set of goals is known at modeling time, goals can change during enactments of informal processes dynamically [DMR15; SF09]. Consequently, the business logic can vary between different enactments of the same informal processes significantly, making repeatability based on interrelated activities aggravating. Moreover, due to changing activities in enactments, it is typically impossible to predict activities and their order at modeling time for informal processes [LR00; DMR15]. Thus, creating activity-oriented business process models of informal processes adds less business value than the effort for creating models. In summary, activities and their order of informal processes are difficult to predict at modeling time and change between different executions of the informal processes aimed at the same goals.

2.1.2 Different Relationships Between Resources

Informal processes typically comprise a team of people supported by a set of resources [DMR15; Dav10]. Being a team implies certain relationships between team members, such as “leading” relationships and “has-worked-together” relationships [RtHEvdA04]. Having such relationships between team members is key for creating coherent teams, as the relationships imply certain team dynamics. Moreover, different roles in teams can result in different levels of clearances, meaning different types of relationships between team members and other organizational resources [DMR15], such as an “update” relationship between a project manager and financial reports. Using such relationships provides a mechanism to manage shared organizational resources among different team members [Dav11]. Thus, relationships can symbolize the control of resources over other resources in business processes. Interestingly, according to property rights theory, the correct distribution of the control of resources (i.e., property rights) to their users has an influence in the efficiency of their use [Bur04; Bur16]. Consequently, the existence of such relationships can increase the efficiency of business processes, as the efficient use of allocated resources will affect the efficiency of business processes. In addition, in certain cases, there are existential dependencies among organizational resources (e.g., an application “depends” on a database). In summary, informal processes comprise various relationships between resources for different purposes, such as (i) social relationships, (ii) clearance relationships, (iii) existential relationships, and (iv) ordering relationships [MWMY11; DMR15; Bre16; Bin15; Dav11].

2.1.3 Multiple Resource Participation in Informal Processes

Organizations build teams with certain structures to address complex organizational problems [MST09]. Organizational teams use other organizational resources in business processes, such as IT resources, to reach the collective goals of the processes [LR00]. To this end, business processes have three dimensions: (i) “what”, (ii) “who”, and (iii) “with” [LR00]. The “what” dimension refers to activities and their required order to accomplish the goals of business processes. The “who” dimension includes every resource, such as IT resources and human actors, performing activities in business processes. Finally, the “with” dimension comprises every supporting resource of the actors. More specifically, information resources holding relevant data for business processes are in the “with” dimension of business processes [EP98; DP98]. In addition, the “with” dimension contains IT and physical resources supporting the completion of the activities [EP98; LR00]. For example, a Wiki service and a simulation device are examples of an information resource and a physical resource, as a Wiki service holds information and a simulation device of a mobile device will support a business process aimed at fixing a defect of a mobile device. Please note that information resources are a special form of physical and IT resources “capable of yielding knowledge” [Dre81; Non94], as the information regarding business processes can be stored both in IT and physical resources. In other words, resources are intangible assets, tangible assets, and human resources providing strengths or weaknesses to an organization, such as the information on a Wiki service, a simulation device, or a Java developer [Bha00; Wer84; Bar91]. To this end, we do not address all types of intangible resources in the scope of resources represented in business processes because of the abstract nature of these resources, such as the reputation of an organization and the trust in it [Bur04; Bha00].

Another kind of intangible resource is capabilities (i.e., skills) of organizations created using other organizational resources [Bha00; Gal05]. Capabilities are the ability to conduct productive activities repeatedly by deploying organizational resources to create value for the organization, such as a software development capability offered by a software developer [Gra96; Bha00; AS93]. Capabilities can be combined into more sophisticated capabilities, such as a product development capability, by combining multiple capabilities provided by different resources or teams [Gra96].

For successful completion of an informal process, all involved organizational resources and capabilities provided by the involved resources play an important role [DMR15; DDB98]. In summary, not only actors of informal processes but also organizational resources supporting these actors are critical for successful process conclusions [Ley12; LR00; DDB98].

2.1.4 Changing Resources

During informal processes, new requirements and consequently new goals emerge and old ones lose their priority [DMR15; HK11]. To adapt to these changes reactively and to resolve certain problems proactively, the set of resources participating in an informal process can be updated dynamically [MWM+12; MWMY11; Non94]. For example, to address a new security issue in a software system, an external expert can be recruited in a reactive manner. In summary, during the life cycle of an informal process, resources participating in an informal process can change.

2.2 Business Process Management

Organizations set and achieve their organizational goals representing collective intents for reaching their desired states from certain initial states [Moh73; dSdSPvS09; RZ94]. Furthermore, organizational goals are typically related to each other through different types of relationships. For example, goals can require accomplishing other goals or can contradict with each other [AMP94; HBJ+14; VCN+01; SGM04; GMNS03]. Moreover, goals can be decomposed into other sub-goals with "and"-relationships or "or"-relationships [ADG09; SGM04; Ant96; VCN+01]. Goal modeling approaches [ADG09; SGM04; DD07; GMNS03] present means of representing goals with their respective relationships in different forms, such as a goal tree [DD07; VCN+01] and a goal graph [SGM04; GMNS03]. A use case of goal modeling approaches is representing software requirements in software development processes [Ant96; SGM04; ADG09]. To this end, a requirement specifies "how a goal should be accomplished by a proposed system" [Ant96]. For example, a goal targeting scheduling a meeting will require the availability of a meeting room at the specified date and time [Ant96].

Operationalizations of goals result in business processes achieving these goals [AMP94; KK97]. Interrelations between organizational goals, business processes, organizational resources, and their capabilities can be modeled using enterprise architecture modeling languages, such as ArchiMate [JLvB+04; Gro16]. Using ArchiMate, it is possible to describe organizational resources that provide capabilities used for certain organizational goals [Gro16]. In addition, business experts can specify business processes for reaching organizational goals. Thus, ArchiMate enables the modeling of architectures of organizations with different layers, such as the strategic layer containing resources and capabilities and the technology layer containing models of IT resources.

Similarly, it is possible to design architectures of human collaborations in organizations. Therefore, business experts can use human Architecture Description Language (hADL) describing architectures of collaborations involving people [DT12a; DT12b]. The language is inspired by software architectures and relies on (collaboration) components and (collaboration) connectors for describing collaborators, messages, streams, shared artifacts, and the relationships between them. Here, collaboration components do work, such as an author writing an article, and collaboration connectors coordinate the work that has been done by collaboration components, such as checking the quality of the work written by authors [DT12a; DT12b].

Business processes go through different life cycle phases [Wes10; DVT05; vdAal16]. In the following, we describe the BPM life cycle of Weske [Wes10], which comprises four phases: (i) design and analysis, (ii) configuration, (iii) enactment, and (iv) evaluation, as illustrated in Figure 2.1.

Design and Analysis The phase of design and analysis involves activities for modeling business processes and analyzing their models. More specifically, during the design phase, business experts conduct surveys to identify business processes. After their identification, they create models of these identified business processes, as shown in Figure 2.1. Validation follows designing business processes. For validation purposes, organizations can organize workshops with different stakeholders participating in the corresponding processes. Moreover, to validate the execution behavior of the designed

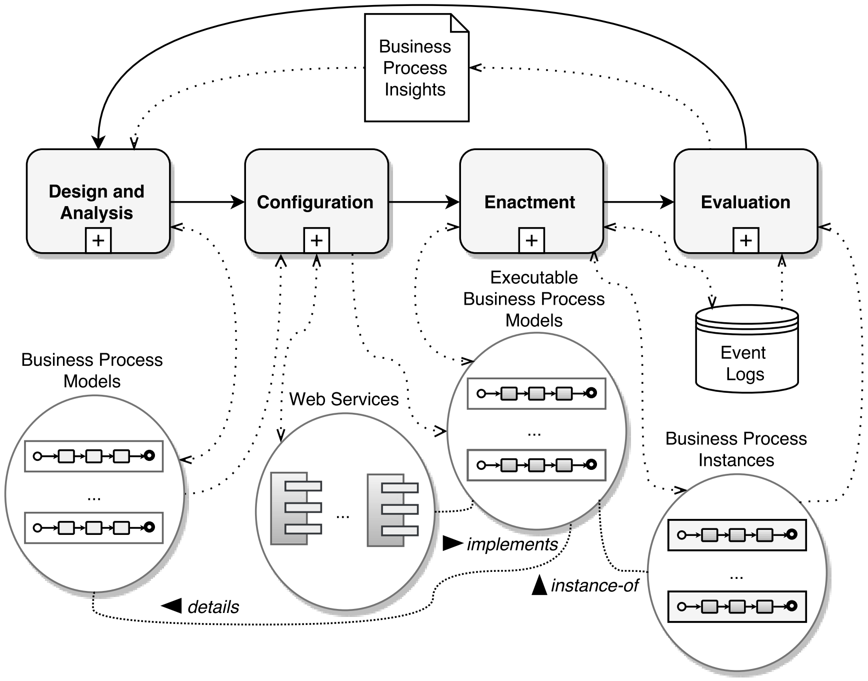


Figure 2.1: Activity-oriented Business Process Management (BPM) life cycle based on the BPM life cycle of Weske [Wes10].

processes, organizations can simulate their execution. Finally, organizations verify their processes by checking certain properties, such as the containment of deadlocks.

Configuration The design and analysis phase is followed by the configuration phase. During the configuration phase, technical experts make business process models executable with the help of the IT infrastructure. Depending on the nature of the considered business processes, different types of implementations are possible. If it is impossible or infeasible to automate activities of business processes, business experts can specify a set of rules, policies, and procedures for business processes. Otherwise, technical experts develop the necessary IT infrastructure capable of automating and monitoring business processes designed during the first phase. Therefore, they select a BPM system and extend this with the necessary web services needed for executing the modeled business processes, as shown in Figure 2.1.

For implementing web services and making business processes accessible within and outside organizations, a service-oriented architectural style is appropriate [LRS02]. Service-oriented architecture contains different roles: (i) service providers, (ii) service consumers, and (iii) service directories. Service providers register their web services with sufficient information for invoking web services at a service directory, such as the format information and protocol information for invoking a web service. For example, technical experts can use Web Service Definition Language (WSDL) for defining the format and protocol information regarding web services [WCL+05]. Service consumers discover services in the service directory, select web services, and invoke these using their web service definitions including the information required for invoking them [LRS02]. Moreover, web services can manage stateful resources, such as a database table. To standardize the management of stateful resources, Web Services Resource [OAS07a] can be used. The Web Services Resource standard exploits a web service standard to address web services (i.e., Web Services Addressing [WCL+05]).

Enactment After configuring business processes, organizations enact these in the third phase. For example, business process execution engines execute activities modeled in activity-oriented business process models. Furthermore, organizations monitor and maintain these during the enactment of business processes. Depending on the approach used for enacting business processes, business experts and human actors participating in business processes can update business process models to react changes. For example, actors can add and remove activities from business process models if additional activities needed during executions. Details of such approaches are explained in Section 2.3. Enacting business processes is followed by their evaluation.

Evaluation During the evaluation of business processes, organizations analyze execution logs of business processes to identify bottlenecks. Therefore, organizations employ process mining techniques, that is, analyzing existing executions of business processes based on event logs to discover, monitor, and improve enactments of business processes [vdAal16; vdAal97; vdAWM04; vdAal11; AGL98]. Typically, it is assumed that software systems produce events that are correlated to certain instances of business processes. Furthermore, event logs contain information regarding executed activities of a specific enactment of a business process. Thus, it is possible to construct a business process model based on event logs of an organization [vdAWM04; AGL98]. Interestingly, Folino et al. [FGP15b; FGP15a; FGP14] presented an approach making no such assumption of correlated events, and used a clustering algorithm to generate business process models based on events created by different collaboration systems.

The analysis of event logs is not limited to the construction of the control flow of business process. For example, it is possible to discover organizational structures and social networks containing different event logs of activity-oriented business processes [VRS05; SvdA08; SGM12; YWLW12; vdAS04]. Furthermore, based on the identified organizational structures and existing organizational knowledge containing information on organization members and their roles, it is possible to determine teams and their characteristics used during the execution of business processes [SCD+16]. Mining event logs of business processes can also be used to generate constraint-based business process models, which specify constraints on business executions instead of describing a well-defined concrete business logic, such as a constraint limiting the execution of an activity by a certain role. For example, Schönig et al. [SCD+16] presented an approach enabling the generation of a constraint-based business process model focusing on organizational constraints, such as rules for resource allocations. Similarly, Ly et al. [LRDR05] presented a means of extracting constraints for allocating resources of activities in business processes based on event logs and the existing organizational database, such as a rule specifying the need for an English-speaking capability for a certain activity.

Instead of focusing on event logs of process-aware information systems, by focusing on interactions occurring in collaborations of organizations, it is possible to do process mining. Correlating interactions with activities during enactments of business processes can enable the prediction of future activities aimed at the same organizational goals. Consequently, based on this correlation, it is possible to recommend activities, as detailed by Dorn et al. [DD11]. Such interactions might be stored in version control systems, such as Git and Subversion [LM12; PCF08]. Furthermore, version control systems can contain certain documents representing well-defined work packages of a project, such as a file representing a project proposal. After associating such files with their work packages, events of version control systems can be investigated to identify activities specific to different work packages [BCM+15]. Interestingly, such event logs also contain interactions between human resources implying the expertise and importance of human resources in certain topics, such as the expertise in legal advice. Consequently, based on interactions of a set of human resources, it is possible to identify the importance and expertise of interaction participants [Sch12; Sch09]. Therefore, Schall [Sch12; Sch09] made use of a ranking model, PageRank [ZAA07; PBMW99], which considers human resources important when there are many incoming messages from other human resources. Creating a team of experts to accomplish tasks requiring certain skills with the least cost is a team formation problem [WZN16]. To this end, minimizing the cost is done by cost functions focusing on certain aspects of a team, such as personal costs, communication costs, and load balancing of experts [WZN16]. Furthermore, Wang et al. [WZN16] analyze a set of algorithms addressing the team formation problem, such as the RarestFirst algorithm [LLT09].

An interaction typically implies a certain relationship between two resources, such as an interaction for creating a document, which results in an ownership relationship. Based on relationships between human resources and information resources[1], it is possible to identify experts for a certain topic [BAdR06]. Begel et al. [BKZ10] presented an approach called Code-

[1]Balogh et al. [BAdR06] refer to these as documents.

book for analyzing interactions between human and information resources involved in software projects, such as source code, files, and bug entries. As a result of applying the approach, different conclusions about the software being developed are possible, such as identifying the people responsible for certain features of the software or dependent on a certain feature. This approach aims at mining business processes using different interactions occurring during enactments of business processes. Moreover, the results of the Codebook approach can be used during the (i) business process analysis and design and (ii) business process enactment phases. More specifically, during enactments of business processes, people can find the right people for the right features. Similarly, business experts can adapt business process models based on the results of process mining. Therefore, after business process mining, business experts evaluate business process models based on the results of the process mining, such as new activity-oriented business processes and organizational structures. After the evaluation phase of a BPM life cycle, business experts redesign their business processes during the first phase, if applicable. For modeling and executing business processes, different approaches exist, such as content-oriented and the activity-oriented modeling approaches. Next, we describe these different approaches introduced for modeling or executing business processes.

2.3 Approaches for Modeling and Executing Business Processes

We present different categories of modeling and executing business processes in organizations. During our categorization, we rely on commonalities among different approaches. Furthermore, the categorization is not strict, as an approach listed under one category can have characteristics of another category. A comprehensive comparison of these approaches with the resource-driven modeling and execution approach presented in this work, based on a set of requirements (Section 3.1), will be presented in Section 3.4.

2.3.1 Activity-oriented Approaches

The business logic of business processes consists of a set of interrelated activities [LR00]. When activities and their order (i.e., control flow) are preserved, organizations can document these activities and their interrelations to reuse these in future executions of the documented business process. Consequently, resulting business process models are based on activities, that is, activity-oriented business process models. Certain resources conduct each activity, such as human resources completing a document. Moreover, according to the definition by Russell et al. [RvdAtHE05], a resource in an activity-oriented business process model is "an entity that is capable of doing work" [RvdAtHE05]. Consequently, resources are human actors and automated IT services responsible for carrying out activities of business processes. Furthermore, adding new resources with an activity-oriented business process model requires updating the definition of activities of the business process model. Although actual people conducting activities may change in each business process enactment, the set of involved roles will be predefined by the set of activities added to a business process model. Thus, allocated resources of business processes are bound to activities specified in the processes. Please note that a comparison between concepts introduced in this work and related work will be presented at the end of respective chapters after explaining different concepts.

Human resources can be allocated for activities based on different priorities, such as skills, workloads, and the allocation history [CGR+13]. Specifically, business experts can specify actors and teams based on different criteria, such as experience and skills, using the Resource Assignment Language (RAL) and its extension RALTeam [CRdR15; CRMR15; CRR12]. To this end, RAL provides a means of specifying a single actor for activities, whereas RALTeam addresses a team with different roles for business processes. Moreover, ArchiMate [Gro16] can also be employed to provide an overview of organizational resources and the associated business processes.

The activity-orientation of business processes results in an easy automation of these processes by automating each activity and executing these in the documented order [Nur08]. Different business process modeling languages are available for defining such activity-oriented business process models, such as Business Process Model and Notation (BPMN) [Obj11], WS-Business Process Execution Language (BPEL) [OAS07b], and Yet Another Workflow Language (YAWL) [vdAtH05]. Because of their activity-oriented modeling constructs, these are well-suited for documenting and automating structured business processes from the business process spectrum presented in Figure 1.1. Moreover, they are suitable for handling predictable exceptions in business processes by providing exceptional paths, such as fault handlers in BPEL and error events in BPMN. Consequently, business experts also use these to model structured business processes with ad hoc exceptions from the business process spectrum presented in Figure 1.1.

To handle unanticipated ad hoc exceptions during runtime, Dadam and Reichert [DR09] presented the Application Development based on Encapsulated Pre-modeled Process Templates (ADEPT) approach enabling dynamic changes on business process models. Business experts can change the business logic of business processes and change parameters of activities designed in business process models to handle these ad hoc exceptions. Similarly, Sonntag [Son16b] presented the model-as-you-go approach, specifically targeting requirements of business processes executed in scientific experiments, such as adapting business processes at runtime [SK10]. The model-as-you-go approach enables different types of adaptations, such as adaptations in the control flow of business processes and repeating and skipping activities. Furthermore, the approach requires the deployment of a partial business process model for further evolution during the execution of the business process [SK10]. Consequently, the model-as-you-go approach becomes well-suited for structured business processes with ad hoc exceptions from the business process spectrum presented in Figure 1.1.

The inclusion of activities conducted by humans is critical for automating business processes involving people. Therefore, BPMN presents (i) user tasks and (ii) manual tasks, representing activities conducted by actors with and without using an IT system. Moreover, BPMN follows a closed-world approach by specifying the set of actors of a business process at design time [BFV11]. Consequently, business processes do not leverage the resourceful information provided by other organization members and possible external contributors at runtime. For example, an activity for analyzing a document can be assigned to a specific role (i.e., closed-world assumption) or can be conducted in participatory way by inviting every member of the organization. To tackle this closed-world approach, social BPMN extensions [FBV11; BFV12; BFVB12; BFV11] shift this closed-world approach to a controlled participatory approach by allowing the participation of additional roles, such as internal and external observers, which are people following the advancement of socially extended business processes affecting these business processes indirectly by making comments and voting, for example.

For specifying human activities in BPEL processes, business experts use WS-BPEL Extension for People (BPEL4People) [OAS10b]. Moreover, the Web Services – Human Task Specification Version 1.1 (WS-HumanTask) [OAS10a] exists for specifying human activities. Using WS-HumanTasks, business experts can specify ordinary tasks and composite tasks containing other sub-tasks. Actors conducting an ordinary task can decide to substructure this task by creating additional ad hoc sub-tasks. Furthermore, during the execution of each task, operational data containing ad hoc attachments and comments store information regarding the execution of tasks. Consequently, each human task specifies its actors and a limited set of information resources. In addition, the specification points out supporting applications without detailing these. Thus, applications supporting human tasks are out of the specification. Holmes et al. [HTZD08] presented a view-based approach to represent different perspectives (i.e., views) of business processes containing related concepts. Specifically, the authors create a human view for business processes and extend this with the concepts of BPEL4People, resulting in a platform-independent meta-model and a platform-dependent meta-model (i.e., a meta-model for BPEL4People) of human aspects of business processes. According to Schall et al. [SDTD09], human tasks defined in WS-HumanTasks define activities in the scope of business processes using the BPEL4People extension of BPEL. To address collaborative situations involving no such activity-oriented business processes models, Schall et al. [SGDD07; SDB10; STD11; STD08; SDTD09] proposed Human-provided Services (HpS) abstracting human capabilities behind interfaces defined using WSDL [WCL+05]. Consequently, actors and software services can offer their functionality in a unified manner. Furthermore, the functionality can be discovered and used by other actors and web services.

When activities of business processes are unpredictable at design time, it is possible to create ad hoc business process models during the process execution. For example, Caramba allows the initialization of business processes without any predefined activities [Dus04; Dus05]. Furthermore, during the execution of business processes, actors add new ad hoc activities. Caramba not only focuses on reusable activities among different executions of the same business processes but also focuses on roles, people, groups, units, capabilities, IT, and information resources to accomplish activities. Consequently, Caramba addresses limited types of IT, human, and information resources. In addition to addressing these resources, business experts can specify named relationships between different resources. Our final research question in Section 1.2.3 addresses the incorporation of resource-driven processes into activity-oriented business processes models. Furthermore, our corresponding contribution in Contribution 5 addresses this research question with the introduction of a new type of activity in activity-oriented business process models. During the development of this new type of activity, we considered the following set of aspects of modeling and executing activity-oriented business processes.

2.3.1.1 Different Aspects of Activity-oriented Modeling and the Execution of Business Processes

Next, we discuss different aspects considered during modeling and execution of activity-oriented business processes. During our analysis, we focused on different features of business processes, such as (i) adaptation and (ii) optimized execution of business processes due to increasing trends toward the automated adaptation and optimized execution of business processes [LFK+14; RLG+15; BKS11; EL16]. Furthermore, we consider two types of adaptations:

(i) based on the information available in the execution environment, that is, a context-sensitive adaptation, and (ii) based on the available methods for reaching the same goals, that is, a goal-oriented adaptation.

Context-sensitive Adaptation Aspects The networked interconnection of things, such as RFID tags, sensors, and actuators, is called the Internet of Things and has opened various new application fields of computing [XYWV12]. The access and manipulation of the state of the real world using networked things enable the creation of environments that can be programmatically operated, resulting in so-called smart environments [AIM10; GBMP13]. For example, smart factories can automatically adapt to changes in their environments [LCW08]. For these adaptations, smart factories observe the information coming from the environment in the business processes and react to these accordingly. In this work, we call such business processes context-sensitive business processes. Moreover, we define context as any information characterizing the situation of an entity, that is, a person, place, or object, which is considered to be relevant for interaction based on the definition provided by Dey [Dey01]. Context-sensitive[2] systems use context during their executions [Dey01]. For example, they provide services and information based on the present context [Dey01]. Consequently, context-sensitive business processes use context during their enactments.

Context is represented as contextual data in IT systems. Moreover, the contextual data in business processes have three different categories: (i) generic, (ii) business process dependent on a priori knowledge, and (iii) business process independent of any a priori knowledge [AtHRvdA09]. The generic contextual data exist for all business processes of an organization, such as the data regarding the start time and state of a business process. In contrast, the contextual data in the category of a "business process dependent on a priori knowledge" is accustomed to the specific enactment of business processes, such as different data variables passed between activities. The last category of the contextual data "business process independent of any a priori

[2]Dey [Dey01] refers to these as context-aware systems

knowledge” addresses the set of data available only at runtime, such as the information regarding a defect in a mobile device. In the scope of this work, context-sensitive adaptation aspects of business processes refer to adapting these based on contextual data. Thus, context-sensitive adaptation aspects of business processes address the specification of the changing business logic based on contextual data available at runtime.

To consider context-sensitive adaptation aspects, in our previous work [SSOK13], we presented a set of extensions to BPMN enabling a set of activities specifically for wireless sensor networks, containing dynamically joining and leaving wireless sensors and actuators [ASSC02]. More specifically, business experts can create BPMN models capable of observing and manipulating the physical world using the sensors and actors of wireless sensor networks. Consequently, business process instances directly interact with wireless sensor networks. Instead of the direct interaction, it is possible to add a context management middleware providing a unified interface for managing context and hiding the management details of wireless sensor networks [FFP12; LCG+09]. Wieland et al. [Wie13; WKNL07; WKN08] made use of such a middleware in their context-sensitive business processes. In their approach, the authors proposed a set of business process modeling constructs for reacting to context-based events and making context-based decisions by collecting data over a context management middleware [WKN08].

Adaptive pervasive flows are business processes logically attached to entities that move with them [BLMP09]. Moreover, adaptive pervasive flows are context sensitive in the sense that they react to changes in their environment. When defining adaptive pervasive flows, business experts specify context-sensitive goals for activities, which are refined at runtime, using available process fragments, that are, reusable incomplete process parts, with certain postconditions [BMPR12]. To model context in adaptive pervasive flows, Wolf et al. [WHR09] presented extensions to adaptive pervasive flows for attaching relevant entities to a group of activities, such as a box being transported in a business processes to an activity for unloading this box. Thus, the attached activities are aware of information characterizing these entities, such as the condition of the boxes. Furthermore, the authors proposed a mechanism for handling changes in the context of associated entities. For example, the proposed mechanism allows the execution of necessary handlers in case a box being transported is broken. Similarly, Adams et al. [AtHRvdA09] presented a business process modeling and execution approach containing context-sensitive activities, that are, activities using contextual data in its execution. The selection of the right business logic for each context-sensitive activity depends on a set of context rules specifying the logic for choosing the business logic during execution. Similarly, Eberle [Ebe14] presented her approach of process building bricks for creating incomplete business process, which are completed at runtime based on the contextual data available.

Mundbrod et al. [MGKR15] presented the Context-aware Process Injection (CaPI) approach, enabling the context-aware dynamic injection of process fragments into previously marked areas in business process models based on a set of rules for injecting these process fragments. As a result, business process models containing these marked areas are not compliant with standards of the corresponding business process modeling language [KBG+16]. Therefore, Képes et al. [KBG+16] presented an approach capable of executing standard compliant business process models in a context-sensitive way. In their approach, the conversion of standard compliant business processes into context-sensitive business processes occurs behind the scenes by selecting the right business logic based on the actual contextual data. Next, we focus on the goal-oriented adaptation aspects.

Goal-oriented Adaptation Aspects Reaching organizational goals requires enacting business processes aimed at these goals. Furthermore, business processes evolve in time, resulting in different variants suitable for executing in different situations. For example, an activity that is manually conducted for assembling a mobile device previously can be done automatically because of technological advancements. Thus, both manual and automated business processes will aim at the same goal with different performance metrics. Moreover, selecting the manual assembly can be reasonable if the assembly machines of mobile devices are not working. The increasing trend of individualizing services, such as production or IT services, typically requires the execution of different process variants based on the requirements of service consumers [LFK+14; HBR10a]. Although different process variants contain a different business logic for executing an individualized service, the organizational goal is the same. During individualization, both (i) the organizational goal, such as producing a mobile device, and (ii) the contextual data, such as size of the memory requested by a customer, are considered. Thus, in the scope of this work, we consider goal-oriented adaptation aspects to achieve organizational goals by considering the available contextual data.

Ontologies are logical theories explaining the intended meaning of a formal vocabulary [Gua98] and are a method of specifying goals. Moreover, ontologies can be defined using different languages, such as Web Ontology Language (OWL) [Bec09]. An example ontology is Web Service Modeling Ontology (WSMO), which presents a vocabulary for specifying web services and enabling the discovery, composition, and invocation of web services [FFST11]. For enabling the context-sensitive dynamic discovery, composition, and selection of web services, Santos et al. [dSdSPvS09] presented a goal-based framework. Therefore, they relied on ontologies describing goals, activities supporting the goals, and corresponding web services implementing these activities. Based on goals of service consumers and goal definitions of activities and web services, the goal-based framework discovers and composes web services. Similarly, Gomez et al. [GRG+04] presented the Goal-oriented Service Discovery (GODO) approach using ontologies for discovering, composing, and selecting web services for a text typed in a natural language. Based on the given subject, verb, and predicate tuples in sentences, GODO selects, composes, and invokes a set of web services. Consequently, queries made in a natural language can be mapped into a set of services satisfying the goals of queries. For example, Salhofer et al. [STSJ08] presented an e-government application of a service discovery based on a query made in natural language. Furthermore, the Semantically AnnotaTed Intentions for Services (SATIS) approach [MC10] enables the discovery of web services required for achieving a set of goals in a given order, such as ordered goals aimed at first preprocessing and then analyzing an image.

Optimized Execution Aspects The performance of business processes is evaluated in different dimensions, such as quality, time, flexibility, and cost [Gla12]. Furthermore, each performance dimension is associated with certain indicators for analyzing these. For example, the cost dimension is measured using manufacturing costs, added value, selling prices, running cost, and services costs [Gla12]. An optimized execution of a business process increases its performance [ZC03]. In this work, optimized execution aspects of business processes are concerned with the optimization of a business process after selecting a business process to be executed based on the current contextual data. For example, removing an activity from a business process is based on the contextual data after the selection of the respective business process aimed at a specific organizational goal. Thus, we do not address selecting an optimized variant aimed at a certain goal during the business process enactment, which is addressed by the goal-oriented adaptation aspects. For example, Hallerbach et al. [HBR08; HBR10b; HBR10a] present the Process Variants by OPtions (ProVOP) approach for dynamically configuring business processes in a context-sensitive way. Therefore, business experts specify a set of context rules, such as a lower cost production for a more sustainable product depending on the contextual data. If context rules specified in business processes evaluate to true based on the contextual data during the enactment, the associated process variant is used, such as a business process without certain activities resulting in a more sustainable product. The optimization problem can be interpreted as a multiobjective optimization problem, due to targeting different objectives during the optimization of business processes, such as shorter time and lower cost objectives. For addressing multiobjective optimization of business process models, Wibig [Wib13] and Vergidis [Ver08] presented algorithmic approaches generating an optimal solution considering different optimization objectives.

The optimization of business process models based on the current contextual data is a planning problem aimed at finding the list of activities to achieve a goal under a certain context [Wel94; HKZ15; HS15]. Expressing a planning problem can be done in different ways, such as using a STanford Research Institute Problem Solver (STRIPS) style planning [FN71]. The STRIPS-style planning expresses a planning problem in terms of a goal describing the desired and current states of the world and a set of actions. The result is a plan containing a totally ordered sequence of actions, which are business process activities. The execution of a plan will transform the current state into the desired state of the world. Heinrich and Schön [HS15] presented such an approach for generating context-sensitive business process models based on the contextual data by describing the business process model as a planning problem. Therefore, the authors formalized different activities of business processes and contexts similar to a planning problem. Each activity in the planning problem has a precondition and effects similar to the abstract activities defined in adaptive pervasive flows [BMPR12; GW96]. Adaptive pervasive flows use the concept of abstract activities, containing no actual implementation, for this purpose [BLMP09]. Business experts describe the target of each abstract activity in terms of the context targeted by the abstract activity. Gómez Sáez et al. [GAH+15] extended BPEL with such an abstract activity. Placeholder activities present a similar concept to abstract activities by introducing underspecified activities detailed during runtime [WRR08; GSBC00]. The actual business logic of a placeholder activity can be selected (i) automatically based on a set of predefined rules or (ii) manually by an authorized actor during execution [WRR08; GSBC00]. Multiple activities of a business process can share the same set of resources, resulting in race conditions during business process executions. For example, allocating the same actor for two simultaneous activities in a business process will typically decrease its execution performance. For avoiding such suboptimal conditions, Havur et al. [HCMP16] presented an approach of scheduling resources of activities in an optimal fashion.

Another style of planning is Hierarchical Task Network (HTN) style planning aimed at describing a planning problem in terms of a set of target tasks with certain constraints, such as ordering and certain preconditions for initializing tasks [EHN94; EHN95; Ero96; GA15]. Furthermore, a HTN introduces two types of tasks: composite and primitive tasks. Composite tasks can be further refined using primitive tasks with certain constraints. The resulting HTN plan is a list of primitive tasks that are activities in the context of business processes [EHN94]. The main difference between the HTN planning and STRIPS planning styles is how the resulting state of the world is represented. The STRIPS-style planning represents this as goal attainment. In contrast, the HTN-style planning describes this in terms of a set of tasks with certain constraints.

Song and Lee [SL13] presented an approach for composing web services to accomplish a certain goal using HTN-style planning. In their approach, business experts specify goals and associate goals with tasks. During the achievement of goals, goals are resolved into a set of tasks. A plan containing a list of activities capable of executing the set of tasks is assembled using HTN-style planning. The approach requires preliminary definitions of goals, tasks, web services, and their relationships using a set of ontologies. Web services can be described using Ontology Web Language for Services (OWL-S) containing a set of ontologies for describing different aspects of web services [Mar+05]. For composing web services defined using OWL-S, Sirin et al. [SPW+04] presented an approach based on HTN planning.

In contrast to relying on ontologies, Lazovik et al. [LAP06] presented the XML-based language, XML Service Request Language (XSRL), to describe service requests. Based on these requests, a plan with its possible executions is generated with a web service composition satisfying the service request.

Similarly, Kaldeli et al. [KLA16; KLA11] presented an approach relying on planning to compose web services. In their approach, they translate the goals of business processes into constraints to be satisfied, which are resolved into a plan containing a service composition considering the initial context and satisfying the constraints. Interestingly, their approach makes use of continual planning to handle situations that cannot be anticipated during modeling. So far, we have detailed approaches focusing on activities and their interrelations for specifying business processes. Instead of focusing on these, it is possible to focus on the constraints needed for enacting certain activities, which we explain next.

2.3.2 Constraint-based Approaches

Organizational rules, policies, and regulations limit the execution of activities in business processes. For modeling business processes based on these constraints, constraint-based approaches describe possible activities and constraints, limiting their execution [RW12]. For example, such a constraint-based approach was proposed by van der Aalst et al. [vdAPS09; PSSvdA07] in their Declare approach. Using Declare, business experts define interrelations of activities of business processes in terms of logical expressions specifying constraints for executing the activities. For example, a constraint can specify that Activity A must be executed every time Activity B has been executed. Declare provides extensible support for languages to specify these constraints. At runtime, business process execution engines supporting the Declare approach evaluate logical expressions of constraints for initializing activities. Upon the initialization of activities, actors select and carry out the activities. Consequently, actors execute only activities whose (mandatory) constraints are fulfilled. Moreover, Declare allows ad hoc changes to business process instances at runtime. The Declare approach focuses on an activity-oriented support for actors by limiting possible activities based on the constraints. In addition to constraint-based ordering of activities, the Context-aware Software Engineering Environment Event-driven frameworK (CoSEEEK) approach aims at the automated selection of activities based on the current context [GOR10; GOR11]. For example, upon the receipt of a defect message, in the current context, the business logic for handling the defect is generated using the constraints among the activities.

Similar to the CoSEEEK approach, van Grondelle and Gülpers [vGG11] proposed an approach limiting possible activities based on preconditions. In their approach, they presented a more restrictive formalism of preconditions. More specifically, preconditions are described in terms of the availability of certain roles, documents, appointments, notes, and objects before initiating an activity. Furthermore, each activity can require a decision to be made before its execution. Business experts can specify time limits for starting and expiring activities, such as preconditions describing an activity that will start in five minutes and should be started in five minutes, respectively. In addition to preconditions, business experts use postconditions to describe the resulting state of the world after enacting activities of business processes. Business experts can specify postconditions similar to preconditions using documents, appointments, notes, and objects. Furthermore, activities can result in decisions, role assignments, and starting, suspending, and resetting time limits of other preconditions.

The approach presented by Grondelle and Gülpers [vGG11] addresses information resources and human resources required at the level of activities. Adding new resources into business process models requires updating preconditions and postconditions of activities. Furthermore, due to focusing on the availability of certain content, the approach becomes similar to content-oriented approaches presented next in Section 2.3.3.

The limiting nature of constraints can be used for adapting business processes executed using activity-oriented business process models to their execution environments. Therefore, Marella et al. [MRM12] presented a "planlets"-based approach with an adaptation mechanism, which are self-contained activity-oriented business process models. In their approach, activities of business process models are annotated with preconditions and postconditions. If preconditions or postconditions are invalidated during the enactment, a recovery procedure is executed, where planlets containing the business logic for bringing the system to a valid state are executed. The planlet-based approach extends activity-oriented modeling approaches with precondition and postcondition annotations. Thus, the presented approach can be used to automate unstructured processes with predefined process fragments from the business process spectrum (Figure 1.1).

Adaptive pervasive flows present a similar approach, enabling the construction of the business logic of a business process at runtime based on preconditions and postconditions[3] [BMPR12; BLMP09]. Constraints can refer to whole business processes [NEK+05] or specific activities of business processes [vGG11; vdAPS09]. For example, Igler et al. [IMZJ10; IMF+10]

[3]Bucchiararone et al. [BMPR12] referred to these as effects

presented the ESProNa approach, focusing on the constraints of whole business processes. Business processes are activated upon the fulfillment of their constraints. Furthermore, the constraints of ESProNa focus on different perspectives of business processes, such as the functional, behavioral, organizational, and data perspectives. Constraints regarding the functional perspective of a business process specify functional behavior, such as how often a business process should be executed. Behavioral perspective constraints specify business processes that a business process depends on before starting its enactment. Organizational constraints describe requirements for human, IT, and physical resources for executing business processes. Finally, data constraints represent the data needed for enacting business processes. Although the authors in one of the works [IMZJ10] claimed that the organizational perspective addressing tools used for completing activities in business processes is supported, there are no further details of this, and they omit this perspective in another work [IMF+10]. Moreover, using the ESProNa approach, business experts can document required IT, human, information, and physical resources indirectly by specifying constraints of business processes. Therefore, business experts define new constraints of business process models associated with new resources.

Opposed to business process modeling and execution approaches focusing on modeling business processes independent from their organizational goals, Nurcan et al. [NEK+05] presented a decision-oriented approach focusing on strategic aspects of business processes. Such models are semantically more powerful due to providing reasoning about business processes of organizations instead of explaining only what needs to be done, such as typical activity-oriented business process models [NEK+05]. To this end, strategies do not define what needs to be done; instead, they have a guiding nature by specifying why organizations enact certain business processes [NEK+05]. Moreover, business processes in organizations serve to achieve organizational goals needed to complete their strategies [Wes10]. Consequently, strategies are decoupled from their operational implementations, which are business processes. Nurcan et al. [NEK+05] proposed a strategy-driven business process modeling approach focusing on organizational strategies and goals.

Business processes are expressed in terms of goals and strategies connecting these goals. For example, a room booking goal is connected to an “accept payment” goal through (i) an electronic transfer strategy and (i) credit card strategy [NEK+05]. Depending on the nature of goals, business experts refine these either (i) recursively with additional goals and strategies or (ii) with business processes describing what needs to be done to reach a goal. Furthermore, business experts specify preconditions and postconditions for business processes. Preconditions specify the conditions to be satisfied to enact the corresponding business processes. Similarly, postconditions describe the conditions after executing the respective business processes. Nurcan [Nur08] claimed that describing the goals for executing a business process in business process models is appropriate for modeling informal processes. Because of the presented preconditions and postconditions, we considered this approach under the constraint-based approaches. Instead of focusing on the constraints of business processes, representing evolving content can be a flexible method to present advancements in business processes, such as representing evolving data objects and files. Next, we present such content-oriented approaches.

2.3.3 Content-oriented Approaches

Content-oriented approaches focus on the content of business processes, representing advancements in business processes. For example, artifact-centric business process modeling is driven based on the state changes of business artifacts, which are explicit information chunks related to business processes [NC03]. For example, an insurance claim in a business process is a business artifact due to the information contained regarding the business process. In other words, a business artifact is a business entity with a state [YL10]. Thus, business artifacts can be considered a subset of information resources due to their business focus.

State changes of business artifacts result in advancements of business processes. For example, the state change to the “approved” state of an insurance claim can result in a service call sending a message to the insurance claimer. To define activities to be executed upon state changes, business experts specify business rules containing preconditions, corresponding services to be called, and postconditions after the execution of services [YL10]. Furthermore, adding new business rules can result in the inclusion of new information resources, which are indirectly business artifacts. The specification of preconditions and postconditions of artifacts limits the possible set of activities. Thus, the approach is suited for modeling (i) loosely structured business processes and (ii) unstructured business processes with predefined fragments in the business process spectrum (Figure 1.1).

Liptchinsky et al. [LKTD12] proposed another artifact-centric business process modeling approach using Unified Modeling Language (UML) state charts of business artifacts. State charts represent different states of business artifacts. Moreover, each state of a business artifact is extended with the context of the respective business artifact. The context contains relevant business artifacts and human resources having an effect on the state of the business artifact. Consequently, this modeling approach can express relationships among different business artifacts, actors, and their states.

Similar to artifact-centric business process modeling approaches, the case handling approach is also categorized as a content-based process modeling approach due to basing the progress of business processes on the evolution of data objects, which are business artifacts in artifact-centric business processes [Wes10; vdAWG05]. Case handling enables the definition of activities involved in business processes, which are cases[4], at modeling time or at runtime in an ad hoc fashion. Furthermore, the initialization of different activities in a case depends on the available data at runtime, such as requiring filled form data to initialize an activity for reviewing the form.

Interestingly, the Object Management Group (OMG) proposed Case Management Model and Notation (CMMN) detailing the concepts introduced in case handling [Obj16]. The CMMN enables (i) modeling activities before the execution of a case and (ii) ad hoc changes during the execution. Furthermore, each case is associated with data objects, such as files and

[4]In case handling terminology, a *case* can be considered equivalent to a business process instances.

folders. Actors of business processes can update the set of associated data objects during the execution of business processes. Using these data objects, it is possible to observe the advancement of a running case and take necessary actions, such as initializing an activity for reviewing the form upon the receipt of data. Each business process can consist of multiple roles with different authorizations, such as a role authorized for adding new activities. Case handling is a transitional step between activity-oriented and content-oriented business process modeling and execution approaches due to inheriting features of both approaches.

Motahari-Nezhad and Swenson [MS13] characterized case management systems supporting CMMN as production case management systems due to their limited degree of support for possible adaptations during enactments of business processes. In contrast, the adaptive case management systems provide more adaptability at runtime for actors [HK11; MS13; MSB+13]. Actors can start with an empty business process model containing no activities and adapt this to requirements of business processes. Moreover, actors can run predefined activity-oriented business processes, update a list of to-dos, and add information resources [HK11]. Thus, it is possible to update the set of allocated resources of business processes during execution. Furthermore, these resources involve information and human resources.

Kumar and Wang presented [KW10] an approach for executing business processes in a resource-based way. Therefore, business experts specify activities and their resource dependencies, such as an activity requiring a data resource to produce a physical resource. Moreover, each dependent activity is initiated upon the availability of the resources. Consequently, the resulting business logic is based on the available resources of an organization. In the scope of this resource-based approach, resources are a set of information, human, and physical resources. The approach requires an initial modeling of activities that are interconnected through their dependencies on resources.

Activity-centered computing presents a central concept of activity shared among actors of business processes [YMMS09; BKHM07; MGM+06; MCF05]. Business experts and actors associate a list of sub-activities, to-do lists, IT resources, information resources, events, human resources with a shared activity. Consequently, activity-centered computing enables documenting IT, information, and human resources needed for executing business processes. Unfortunately, the approach does not address relationships among these resources. Interestingly, the concept of to-do lists provides a guide for actors without restricting their actions. Actors can collaboratively work on the shared activities containing to-do lists and update these based on the progress of activities. In the scope of this work, activities presented in activity-centered computing are equivalent to business process models. Automating business processes requires a capable IT infrastructure. To this end, cloud computing provides promising advantages to organizations, which we explain next.

2.4 Cloud Computing

Cloud computing enables the on-demand and measured access to a pool of computing resources over a network without requiring human interaction [MG11]. As a result, computing resources becomes a utility for organizations, such as electricity and the Internet, which enable new business models [Ley09]. More specifically, organizations can go into business without making capital expenditures, such as buying new hardware for their IT infrastructures, as they can use cloud service providers for their IT operations [Ley09; MLB+11].

There are different service models offered by cloud providers, which include people, entities, and organizations making web services ready for cloud consumers [BML+11]. For example, cloud providers can offer software as a service by enabling access to an application and its functionality. In the case of software as a service offerings, cloud consumers do not have control over the underlying details of an application, such as network, memory, and the operating system used by the software itself. In contrast, platform as a service offerings enable the selection of platform features, such as the type of the operating system and its libraries, on which custom applications can be deployed and used. Finally, infrastructure as a service provides the means of allocating, processing, networking, storing, and other computing resources to cloud consumers. To create a pool of computing resources, cloud providers leverage virtualization technologies abstracting their hardware resources, resulting in virtual machines, such as VMware, Xen, and KVM [BDF+03; Wal99; Hab08; ZCB10; Ley09]. Furthermore, in addition to computing resources provided by hardware resources, the virtualization technologies can be used to abstract computing power provided by human resources [DT12c; DB11; CTD13]. For example, the computing power provided by humans is used for evaluating the quality of multiscale simulations, which are simulations containing unmerged mathematical models [WAHK15; DT12c].

On top of virtual machines, cloud consumers can install guest operating systems with full functionality of operating systems and deploy their applications inside of these operating systems. In contrast, to avoid the unnecessary resource consumption of guest operation systems, cloud consumers can replace these with lightweight virtualization technologies, such as Docker [LFWW16; Tur14]. Docker users can distribute their applications using Docker images. After deploying images containing applications, applications run in their containers in an isolated fashion. Moreover, for deploying applications, involving multiple Docker images, Docker Compose can be used to orchestrate the deployment [Tur14].

The on-demand use of computing resources requires adapting the resources for cloud consumers based on their needs [VKL13]. Thus, one of the key characteristics of cloud computing is elasticity, that is, automatically increasing and decreasing computing resources, such as the amount of computing power for cloud consumers, based on their use [Ley09; MG11]. Leveraging different characteristics, such as elasticity and on-demand self-service of cloud computing, requires certain types of architectural properties from the applications being deployed on cloud providers. For example, putting an application into one virtual machine will not utilize the elasticity characteristic of cloud computing, as it will not be possible to replicate individual components and will result in new separate applications instead of copies capable of working on a distributed workload [LFWW16].

Therefore, Fehling et al. [FLR+14] presented a set of patterns, that are documented proven solutions to recurring problems [FBB+14], for building native cloud applications. For example, one of the patterns for creating native cloud applications is loose coupling, that is, adding a communication middleware for separating the application functionality from concerns of communication partners in terms of their location, the platform, the time of communication, and the message format [FLR+14]. Furthermore, for building such loosely coupled applications, Hohpe and Woolf [HW04] presented enterprise integration patterns. Following a loosely coupled architectural style results in the capability of increasing the number of different individual application components, such as the data access layer, of an application for handling different kinds of workloads, which is required for leveraging the elasticity of cloud computing [LFWW16].

Another aspect of elasticity is the automated management of computing resources based on-demand changes, that is, the automated allocation and deallocation of computing resources. For example, to increase the number of web services doing a computational task, it may be necessary to create a new virtual machine in a cloud provider, install a servlet container, and deploy the web service on it. To automate the management of such resources (i.e., creating and terminating computing resources), organizations can create executables, such as business process models and Bash scripts [WASL13]. Moreover, a cloud consumer can desire to switch their cloud providers due to different benefits and limitations provided by different providers, such as different pricing models of cloud providers [BYV+09; ZCB10]. Such a change will require the executables managing the application to be portable between different cloud providers, so that cloud consumers can carry their applications. To tackle the issue of the portability of cloud applications and to provide further benefits, such as interoperability, OASIS [OAS13b] introduced the Topology Orchestration Specification for Cloud Applications (TOSCA) [BBLS12]. In the next section, we explain the details of TOSCA.

2.4.1 Topology Orchestration Specification for Cloud Applications

TOSCA enables the portability of cloud applications by providing a standard for defining cloud applications. Therefore, cloud application developers create service templates specifying their cloud applications. For example, they can use the web application Winery for creating service templates [KBBL13]. Moreover, creating service templates requires the definition of topology templates, which are directed graphs, describing application topologies consisting of different application components and their relationships. Application components refer to individual components of applications providing certain functionalities [BBKL14a]. Figure 2.2 presents an example topology template of a MediaWiki application based on the Visual notation for TOSCA (Vino4TOSCA) [BBK+12]. To define interrelated application components, TOSCA presents a type system consisting of node types and relationship types and their instances (i.e., node templates and relationship templates). To this end, node types represent reusable and customizable application components, such as “Ubuntu” node types in Figure 2.2. Furthermore, each node type specifies the possible capabilities and requirements of an application component, such as a Bash environment capability provided by an “Ubuntu” node type and a relational database requirement of an application. Similar to node types, relationship types facilitate the creation of reusable relationships between application components, such as “hosted-on” relationship types in Figure 2.2. Relationship types can connect either certain node type pairs or capability and requirement pairs. Furthermore, node types and relationship types provide a means of customization by their property definitions, such as inputting the memory size of an Ubuntu machine.

During the design of application topologies, cloud application developers add node templates into topology templates of service templates and connect the node templates with relationship templates, such as the “Ubuntu-1” node template in Figure 2.2. After adding node templates and relationship templates, cloud application developers customize these based on their properties, such as setting host names of Ubuntu machines and their memory sizes. In the case of incomplete topology templates, it is possible to complete

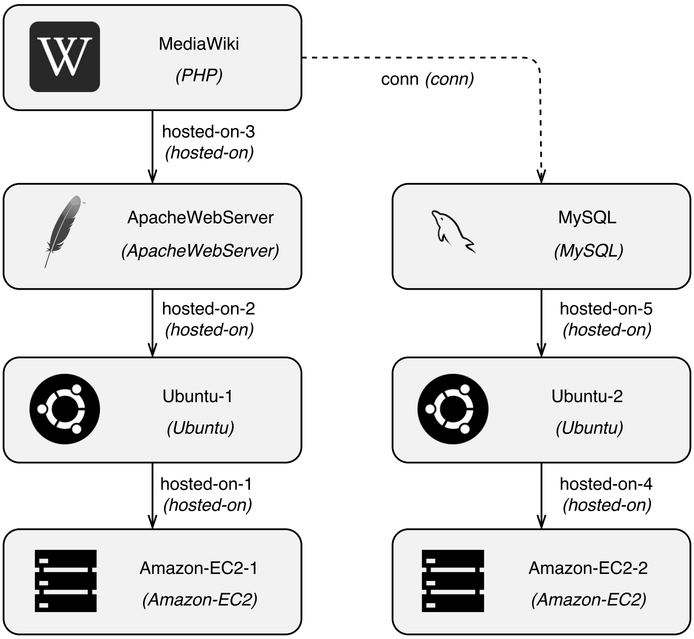


Figure 2.2: A sample topology template of a MediaWiki application based on Vino4TOSCA representation [BBK+12].

them automatically based on the specified capabilities and requirements in node types and relationship types [HBBL14].

Topology templates describe the structure of an application and their interrelations for allocating and deallocating the application using declarative and imperative approaches [BBK+14; BBKL14b]. In declarative approaches, it is sufficient to specify the structure of an application without describing the business logic for allocating and terminating the application. Hereafter, a declarative runtime supporting the TOSCA specification allocates an application by analyzing its topology template. Consequently, the business logic for allocating and deallocating an application is shifted to the corresponding runtime environments. In contrast, imperative approaches require the definition of additional initialization and termination plans, which are executables referred to by service templates, such as Bash scripts and BPMN processes. For example, Kopp et al. [KBBL12] presented extensions for BPMN enabling the creation of initialization and termination plans using BPMN. Node types describe the set of interfaces, providing a set of operations that can be executed on the specified application component, such as a life cycle interface of an Ubuntu node type containing operations for allocating and deallocating an Ubuntu machine. Moreover, for implementing these interfaces of node types, cloud application developers create implementation artifacts, such as a web service, realizing the life cycle interface of Ubuntu node types.

Another type of artifact in TOSCA is a deployment artifact representing the materialization of an application component, such as a virtual machine image. Thus, deployment artifacts are deployed in the environment of the initialized cloud application [OAS13a; OAS13b]. Initialization plans of service templates describe activities to initialize a cloud application and use operations made available by implementation artifacts. Moreover, the execution of these operations can lead to the deployment of deployment artifacts. Consequently, imperative approaches explain how a cloud application must be deployed and increases flexibility and customization options. In contrast, a declarative approach specifies only the topology template for a cloud application, resulting in less flexibility. The topology will be deployed based on the types of nodes and relationships and their attributes. Furthermore, a hybrid approach will generate an initialization plan using types of node and relationship templates added in a topology template [BBK+14; BBK+16].

After completing the definition of a service template and the implementation of its associated artifacts, cloud application developers create an archive file called Cloud Service ARchive (CSAR). The runtime environments supporting the TOSCA specification, which are TOSCA containers, are capable of (i) parsing these CSAR files and (ii) deploying the specified application. The OpenTOSCA Container is an example of such a TOSCA container following an imperative style [BBH+13]. After initializing service templates, each running application can be represented with an application instance model, that is, an application topology containing the additional instance information, such as the IP address, of applications. Binz et al. [Bin15; BBKL13] presented a meta-model for describing application instance models. Similarly, Breitenbücher [Bre16] presented a declarative management language for changing the state of application instance models, such as migrating an application from one cloud provider to another.

Chapter 3

Resource-driven Processes

Business processes are not only structured business processes, such as manufacturing processes but are also informal processes, such as that aimed at fixing a defect of a mobile device, as explained in Section 1.1. Furthermore, the increasing trends toward individualizing products and services raise the need for supporting actors of informal processes and for reproducing their desired outcomes. To address this rising need, we present a resource-driven process modeling and execution approach, answering the research question explained in Section 1.2.1 and resulting in Contribution 1.

Next, we present a set of requirements for supporting actors of informal processes and for reproducing their results (Section 3.1). In Section 3.2, we explain the concept of resource-driven processes that meet the requirements introduced in the following section. To detail the concepts of resource-driven processes more precisely, we present the Resource-driven Process Modeling Language (Redo), a formal language for creating definitions of resource-driven processes, in Section 3.3. After presenting Redo, we continue with an evaluation of resource-driven processes using the requirements in Section 3.4. Finally, we present a discussion of our resource-driven process modeling and execution approach.

3.1 Requirements for Supporting Actors of Informal Processes and for Reproducing Their Desired Outcomes

Based on the properties described previously in Section 2.1 and a set of interviews we conducted to investigate informal processes [Sie15], we derived a set of requirements for supporting actors of informal processes and for reproducing their desired process outcomes. Thus, the following requirements are not a complete set of requirements but present key aspects for this aim.

3.1.1 Base Assumption of Unstructured Business Processes (Rq_1)

Changing activities among different enactments of informal processes makes capturing reusable activities and their structure of processes impossible. Consequently, reusing activities of informal processes provides limited support for actors of informal processes. Moreover, conducting documented activities can fail to reproduce desired process outcomes, as activities change from process enactment to process enactment. Therefore, the absence of interrelated activities repeated among different process enactments requires an approach capable of (i) providing support for actors of informal processes and (ii) reproducing desired outcomes from different enactments of similar processes without relying on reusable activities. As a consequence, an approach to support actors of informal processes and to reproduce desired outcomes should be able to work in the absence of predictable and repeated activities (derived from Section 2.1.1).

3.1.2 Resource Relationship Definition (Rq_2)

Informal processes involve organizational resources that are connected to each other with different types of relationships. Having these relationships is important for different aspects, such as for performance, security, privacy, and functionality. For example, certain social relationships, such as “leading” and “has-worked-with”, among different team members can result in coherent teams performing better than with teams with no such social relationships. Moreover, relationships between team members and other organizational resources can restrict undesired access to different resources and can increase the security and privacy of an informal process enactment, such as limiting the access to the customer information increasing the privacy. Finally, requiring the existence of certain organizational resources before allocating other resources results in a functioning resource set. For example, a team cannot be expected to function properly before receiving the data. Missing relationships can result in decreased performance, security, privacy, and functionality. Clearly, performance, security, privacy, and functionality improvements provided by relationships are important for supporting actors of informal processes and for reproducing their desired outcomes. Furthermore, establishing many relationships requires an additional configuration effort on different resources, such as limiting the access to the customer information through configuring an access management system. Automating these configuration activities will increase the support for actors by decreasing the number of unproductive activities. Thus, an approach to supporting actors for informal processes and reproducing their desired outcomes must provide a means of (i) documenting relationships between resources and (ii) establishing the relationships during the enactment of informal processes automatically (derived from Section 2.1.2).

3.1.3 Resource Visibility Definition (Rq_3)

Informal processes involve not only actors but also various organizational resources supporting these actors during informal processes. For example, the knowledge created in an enactment of an informal process can be valuable and reused during the future enactments of similar informal processes. Moreover, IT tools to create, disseminate, and apply knowledge are typically required to complete informal processes successfully. Finally, depending on the nature of an informal process, physical resources can become more important, such as a maintenance process for fixing production machines manually. Obviously, representing resources in definitions of informal processes will increase the support for actors, as it will reduce the time needed to search for resources to complete informal processes, such as the software to analyze a dataset and the dataset itself. Furthermore, documenting resources will increase the probability of reproducing the desired outcomes of informal processes. Thus, an approach to this must represent different categories of resources: (i) human, (i) IT, (iii) information, and (iv) physical resources, in definitions of informal processes (derived from Section 2.1.3).

3.1.4 Support for Dynamically Changing Resources (Rq_4)

Because of changing requirements of informal processes, business experts and actors can update the set of resources in informal processes during the execution of informal processes, such as adding new software for analyzing data, and the data itself. Documenting these resources can increase the support for actors of informal processes by presenting the additional resources included in previous executions, such as presenting a relevant file added during the previous execution. As a result of documenting resources needed during the execution of informal processes, not only the resources considered at modeling time but also all resources actually required for completing an informal process will be listed in definitions of informal processes. Furthermore, documenting all resources for an informal process will increase the chance of reproducing desired outcomes. Thus, an approach to supporting actors of informal processes and reproducing their desired outcomes should provide means of changing resources in definitions of informal processes dynamically at runtime (derived from Section 2.1.4). Next, we present the concept of resource-driven processes meeting these requirements.

3.2 Resource-driven Processes

To address the requirements listed previously, we propose the concept of resource-driven processes capable of supporting actors of business processes and reproducing their desired outcomes. This section highlights important characteristics of resource-driven processes in an abstract manner. Furthermore, we present formal definitions of the introduced concepts in Section 3.3.

The approach is centered on essentials[1] of business processes, which are resources. Moreover, a resource-driven process is defined as follows:

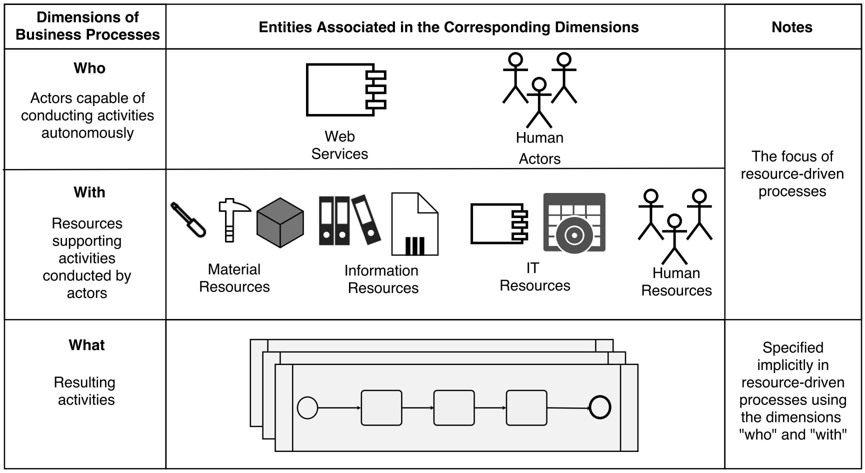
Definition 2 (Resource-driven Processes). Resource-driven processes represent business processes (i) modeled using their interrelated resources and (ii) executed by automatically allocating the interrelated resources for meeting the business process goals, if applicable. ♣

An approach aimed at supporting actors of informal processes and at reproducing their desired outcomes should not solely rely on predictable and repeating activities of informal processes, as stated in requirement Rq_1 (Section 3.1.1). Interestingly, resources of informal processes are the most fundamental requirements for supporting actors of informal processes and carrying these to successful conclusions. Thus, instead of modeling informal processes based on activities and their interrelations, we propose to model them based on the resources required for successfully finalizing them.

Modeling resources will result in two types of support for actors of business processes. First, actors will find resources easily by checking definitions of resource-driven processes, such as a specific Java developer for fixing a defect of a mobile device. Second, documented resources will support actors of informal processes, such as the support information contained in a documented Wiki service. Furthermore, instead of automating the coordination of interrelated activities and their execution as in activity-oriented approaches, we propose to automate the allocation of interrelated resources capable of generating the necessary business logic for completing business processes. Consequently, the resource-driven process modeling and execution will provide means of not only supporting actors but also reproducing their desired outcomes by allocating actors capable of driving processes to successful conclusions.[2] Such an approach will be capable of modeling every kind of business process from the spectrum (Figure 1.1) including structured

[1]Therefore, in our previous publications, the approach was called informal process essentials.

[2]We validate these claims in Chapter 6 with a survey.



Dimensions of Business Processes	Entities Associated in the Corresponding Dimensions	Notes
Who Actors capable of conducting activities autonomously	Web Services; Human Actors	The focus of resource-driven processes
With Resources supporting activities conducted by actors	Material Resources; Information Resources; IT Resources; Human Resources	
What Resulting activities		Specified implicitly in resource-driven processes using the dimensions "who" and "with"

Figure 3.1: Focus of resource-driven processes based on the dimensions of Leymann and Roller [LR00].

business processes and those with ad hoc exceptions, as every business process depends on certain resources. Thus, we use the more generic word "business process" instead of "informal processes", unless it is ambiguous. As shown in Figure 3.1, resources considered in resource-driven processes are not only the "who" dimension (actors) of business processes but also the "with" dimension: IT, physical, and information resources. Modeling the "who" dimension ensures documenting organizational actors capable of autonomously creating activities during enactments of the business process. Moreover, modeling the "with" dimension ensures actors have all organizational resources to meet the business processes' goals. Thus, resources addressed in the "with" dimension provide support for the resources of the "who" dimension, such as a Wiki service supporting actors with the information collected in the service. Furthermore, in the context of resource-driven processes, we define organizational resources as follows.

Definition 3 (Resources in Resource-driven Processes (informal)). Resources of resource-driven processes are every intangible asset, tangible asset, and human resource, such as information, IT, and physical resources, affecting outcomes of business processes in organizations. Their existence does not depend on the existence of other resources. ♣

The presented organizational resource definition is a comprehensive definition including all organizational resources, such as human, information, IT, and physical resources affecting business processes. For example, according to the definition of resources, business processes modeled in an activity-oriented fashion are resources in resource-driven processes, too. More specifically, an executable manufacturing process defined in the Business Process Model and Notation (BPMN) [Obj11] will be considered an IT resource. In addition, a non-executable business process documenting a defect registration process defined in BPMN will be considered an information resource. Consequently, resource-driven processes enable the benefits of activity-oriented business process models implicitly. Moreover, the definition in Definition 3 points out a fundamental assumption of the resource-driven process; each resource contained in the definitions of resource-driven processes must exist in a standalone fashion. For example, the MediaWiki node template in Figure 2.2, requiring a separate database, is not considered an organizational resource, but the MediaWiki service template, with all its dependencies, is a resource in resource-driven processes. The reason behind this assumption is delimiting resource-driven processes from modeling languages describing the structure of applications, such as Topology Orchestration Specification for Cloud Applications (Section 2.4.1) and ArchiMate (Section 2.2).

During the execution of business processes, organizations typically deploy multiple resources. Moreover, to make resources work coherently with each other, there is a need for specifying how they work with each other, as explained in Rq_2 in Section 3.1.2. For example, a project manager should have access (i.e., a “managing” relationship) to a repository of private financial documents of a project and others should not. To represent such explicit relationships, we included the concept of relationships between resources in the resource-driven processes. Relationships are specified as follows in the context of resource-driven processes.

Definition 4 (Relationships in Resource-driven Processes (informal)). Relationships of resource-driven processes are relations between two resources prescribing (i) historical facts of the resources and (ii) desired facts of the resources during enactments of business processes. ♣

In Definition 4, we address two kinds of relationships. The first refers to relationships based on the information existing before the initialization of resource-driven processes, such as a “friends-with” relationship, a “has-led relationship”, and a “has-worked-together” relationship. In contrast, the second type is established after the allocation of resources, such as an “update” relationship between a project manager and a financial repository, a “use” relationship between a service and a database, a “managing” relationship between a project manager and a project database. Relationships referring to the past of resources require checking existing information about the resources to validate the satisfaction of such relationships. Moreover, relationships referring to desired facts of the resources can require executing additional operations, such as adding a new user to a repository.

Organizations employ their resources to create their organizational capabilities as explained in Section 2.2. Therefore, capabilities can be considered an abstraction layer over resources. Using such an abstraction enables decoupling from actual resources, such as a capability abstracting a set of resources capable of developing a mobile device. Furthermore, we define capabilities in the scope of resource-driven processes as follows:

Definition 5 (Capability of Resources (informal)). Capabilities represent abilities to perform activities serving to goals of business processes by deploying organizational resources. Capabilities are for documentation purposes and do not have operational semantics. ♣

Allocating resources with the right capabilities in resource-driven processes should result in the successful achievement of goals. Capabilities can represent abilities for performing single productive activities, such as a capability for programming in Java. Interestingly, organizations can combine such capabilities, referring to single tasks to reach higher levels of capabilities, such as a new service development capability involving a project management capability and a project coordination capability, as described in Section 2.2. Thus, capabilities can be defined recursively based on other capabilities.

Modeling organizational interrelated resources without offering a collective intent is not sufficient to conclude their enactments successfully because resources themselves typically pursue their subjective goals or do not have any goals. For example, employees in a company typically do not work on the goals of the company unless managers engage the employees in the goals explicitly. To engage resources allocated for a business process, we include goals in definitions of resource-driven processes. Consequently, goals guide and limit the activities of autonomous actors of business processes. In the context of resource-driven processes, we define a goal as follows.

Definition 6 (Goal in Resource-driven Processes (informal)). Goals are specifications of collective organizational intents for reaching desired outcomes under specific contexts using certain capabilities. ♣

Defining collective intents of resources guides activities conducted by actors of resource-driven processes. Consequently, goals provide declarative means of specifying business processes by describing expected outcomes without specifying the means of reaching them. Furthermore, goals not only explain outcomes after achieving the goals but also specify under which circumstances they need to be achieved. For example, a goal aimed at fixing a defect of a mobile device should be dependent upon the presence of a defect. The presence of the defect represents the initial state of the environment for achieving the goal. Clearly, achieving goals requires certain capabilities.

Associating goals with capabilities instead of the resources providing these capabilities isolates goals from the changes in organizational resources, such as a Java development capability isolating from actual Java developers of an organization. Therefore, changes in organizational resources do not influence goals. Furthermore, using capabilities instead of resources provides flexibility to organizations by enabling the achievement of the same goals with different resources, resulting in different performance metrics, such as the resulting quality and time (Section 2.3.1.1). For example, (i) machines and (ii) humans providing a mobile device packaging capability for a business process will typically result in different performance metrics.

Instead of using different resources with the same capability for achieving goals, organizations can enact different business processes requiring completely different capabilities. For example, fixing a defect of a mobile device can be done (i) in house, or (ii) the problem can be outsourced by hiring experts capable of fixing the defect, resulting in two different business processes with different capabilities, such as (i) a Java development capability and (ii) a recruitment capability, respectively. Goals indicate all possible capabilities relevant for reaching desired outcomes and do not limit the capabilities to those employed in a single business process. To this end, desired outcomes of goals refer to states after accomplishing the goals. In the scope of this work, we describe different states of organizations using the term context.

Definition 7 (Context in Goals (informal)). Contexts are specifications of certain states in organizations. ♣

By documenting the context of goals, business experts can easily identify relevant goals for current situations. Furthermore, based on their target context, they can select an appropriate goal for this context. For example, the receipt of a report regarding a defect of a mobile product will match the initial context of a goal to fix a defect of the mobile device. Furthermore, its final context will indicate a mobile device functioning properly without defect. The means of specifying contexts is not in the scope of this thesis and can vary from one organization to another. For example, definitions of contexts can be plain text or in a machine-processable format, such as Extensible Markup Language (XML) [Wor98], to facilitate the automated recognition of contexts, such as defining the presence of a defect in a mobile device in a machine-processable format for automating its recognition. To achieve goals, organizations allocate their resources, which requires determining a set of resources capable of achieving the goals. Therefore, organizations use resource-driven process definitions documenting interrelated resources required for goals. Resource-driven process definitions are defined as follows.

Definition 8. (Resource-driven Process Definition (informal))
Resource-driven process definitions document interrelated resources for achieving the goals of business processes. Upon the initialization of resource-driven process definitions, documented interrelated resources are automatically allocated to the business processes, if applicable. ♣

Resource-driven process definitions can be initialized to accomplish organizational goals in a resource-driven fashion. Goals in resource-driven process definitions guide allocated resources by providing information regarding the desired outcomes of business processes during their execution. Furthermore, capabilities required by goals implicitly describe the distribution of allocated resources of a resource-driven process to the goals. For example, if a goal contains a capability of programming in Java, an allocated resource with a matching capability will work on this goal. Capabilities specified in goals are for documentation purposes and do not have operational semantics, as they do not necessarily match the actual capabilities required by the goals of the resource-driven process. Upon the initialization of resource-driven process definitions, enactments of business processes start by automatically allocating interrelated resources. During enactments, actors or business experts can update the set of resources in resource-driven process definitions, triggering automated allocation and deallocation of new and old resources. Therefore, it is important that available organizational resources are visible to actors and business experts updating resource-driven process definitions during executions. Further details regarding modeling and executing resource-driven processes are covered in Chapter 4. Next, we present a formal language for creating definitions of resource-driven processes, as introduced before.

3.3 Resource-driven Process Modeling Language

Documenting business processes in a resource-driven way requires a modeling language capable of creating resource-driven process definitions based on different entities, such as their desired outcomes, resources, and capabilities. Therefore, in this section, we present the Resource-driven Process Modeling Language (Redo), that is, a formal language for creating resource-driven process models, which are resource-driven process definitions. Next, we give an overview of Redo. In Section 3.3.2, we present a set of base elements shared among different language constructs, such as definitions of identifiable entities providing a shared basis for definitions of contexts and capabilities. The explanation of core elements is followed by the description of modeling elements of Redo in Section 3.3.3.

3.3.1 Overview of Redo

Redo presents different modeling elements for creating resource-driven process definitions. Moreover, resource-driven process definitions are stored under an organizational definition along with other kinds of types and definitions, such as resource and relationship types and definitions of goals, capabilities, and contexts. Organizational definitions are the root elements containing the rest of the language components, including resource-driven process definitions. More specifically, organizational definitions contain a set of top-level modeling elements (Section 3.3.3) comprising a set of top-level base elements (Section 3.3.2), as illustrated in Figure 3.2. Formally, organizational definitions are defined as follows:

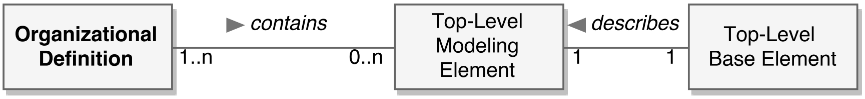


Figure 3.2: An overview of classes and relationships of Redo.

Definition 9 (Organizational Definition). Let an organizational definition be $od \in OD$, such that OD is the set of all organizational definitions.

Each organizational definition is a seven-tuple:

$$od = (ie, \mathcal{S}_{od}, \mathcal{R}_{od}, \mathcal{C}_{od}, \mathcal{G}_{od}, CT_{od}, \mathcal{P}_{od})$$

Where:

- ie is an identifiable entity definition specifying od (Definition 15),
- $\mathcal{S}_{od}$ is the set of all resource types in od (Definition 21),
- $\mathcal{R}_{od}$ is the set of all relationship types in od (Definition 23),
- $\mathcal{C}_{od}$ is the set of all capability definitions in od (Definition 22),
- $\mathcal{G}_{od}$ is the set of all goal definitions in od (Definition 28),
- CT_{od} is the set of all context definitions in od (Definition 27),
- $\mathcal{P}_{od}$ is the set of all resource-driven process definitions in od (Definition 29). ♣

Organizational definition represents the root container of all definitions required to create resource-driven process definitions in a single organization. Moreover, it contains a set of top-level modeling elements, such as resource and relationship types and definitions of goals, capabilities, and contexts. The top-level modeling elements are a part of further modeling elements, as detailed in Section 3.3.2. Furthermore, modeling elements of the language are defined using a set of base elements (Section 3.3.2). For the sake of clarity, we use the subscript od on the set names to address a set containing elements defined under an organizational definition or under an element contained in an organizational definition, such as resource types $\mathcal{S}_{od}$ and identifiable entity definitions IE_{od} representing all identifiable entity definitions contained in an organizational definition and other elements contained by it.

In the following definitions, we refer to domains and range of functions mapping different tuples with different elements. To ease the comprehensibility of these, we introduce the following functions. The domain of a function is the set of possible input values [Sip12]. A domain function is used to find domain sets of another function and is specified as follows:

Definition 10 (Domain Function). Let $f : X_1 \times .. \times X_k \rightarrow Y$,

$$dom_i(f) := X_i, 1 \leq i \leq k$$

The presented domain function inputs a multivariable function, resulting in a different domain for each variable. To reflect this changing domain for each input parameter, the domain function uses index mapping to the domain set of the input parameter at the given index. When a function has a single variable, the index (i) of the domain function can be omitted. The set of possible output values of a function is called the range of the function [Sip12]. To identify the range of functions, we use the following function:

Definition 11 (Range Function). Let $f : X_1 \times .. \times X_k \rightarrow Y$,

$$img(f) := Y$$

The range function returns a set representing the possible return values of a function.

Redo contains various tuples, such as definitions of goals and resource-driven processes. Moreover, we refer to different elements of these tuples. To ease the understanding of our explanations regarding the tuples, we present a projection function returning an element at a certain index in a tuple:

Definition 12 (Projection Function). Let a projection function be:

$$\pi_i(x) : X_1 \times X_2 \times .. \times X_i \times .. \times X_k \rightarrow X_i$$

$$(x_1, x_2, .., x_i, .., x_k) \mapsto x_i$$

The projection function is useful for describing elements of tuples in different definitions. Furthermore, different elements of Redo contain shared properties, such as goals and capabilities, sharing a name and namespace pair identifying these in an organizational definition. Based on shared properties, we created a set of base elements, as shown in Figure 3.2. Next, we detail

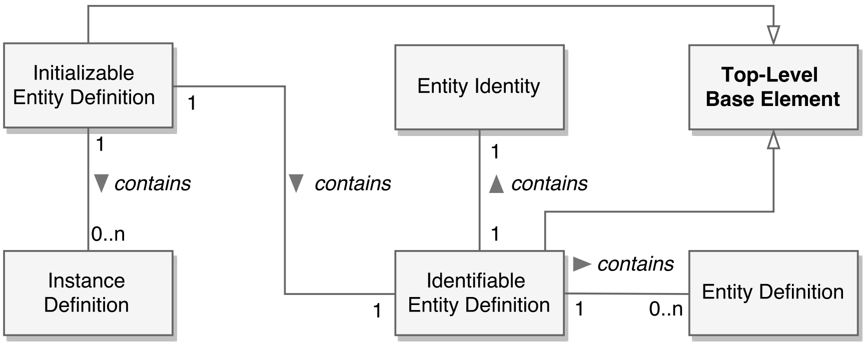


Figure 3.3: Classes and relationships of base elements of Redo.

the base elements of Redo for creating modeling elements of the language, which will be detailed in Section 3.3.3.

3.3.2 Base Elements of Redo

Base elements shown in Figure 3.3 are used to describe modeling elements shown in Figure 3.4, such as context definitions and goal definitions. Furthermore, base elements are used in different modeling elements of the language. For example, every element of Redo includes definitions describing the element itself, such as a definition specifying the context of the presence of a mobile defect in human-readable format. A formal definition of modeling elements is presented in the next section in Definition 18. Moreover, different organizations can specify different elements in Redo using different formats, such as context definitions in human-readable formats or in machine-processable formats. Therefore, Redo should provide a method of creating definitions meeting different requirements of organizations, such as the automation requirements, resulting in definitions in machine-processable formats. By providing these means, the language can be easily extended with additional definitions required by organizations. To enable such extensible definitions of different elements of Redo, we introduced entity definitions as follows:

Definition 13 (Entity Definition). Let an entity definition be $ed \in ED$. Each such entity definition is a pair:

$$ED \subseteq \Sigma^{+} \times \Sigma^{*}$$

$$ed = (type, definitionContent)$$

Where:

- $\pi_1(ed) = type$, $type \in \Sigma^{+}$ is a globally unique nonempty string specifying the type of entity definition,
- $\pi_2(ed) = definitionContent$, $definitionContent \in \Sigma^{*}$ is a string defining an entity. ♣

The entity definition describes different modeling elements of Redo in a certain definition type, such as BPMN and Topology Orchestration Specification for Cloud Applications (TOSCA) (Section 2.4.1). Therefore, the type of an entity definition provides a unique string to identify the content stored in the entity definition. To this end, Σ is the set of unicode symbols. Entity definitions are elements within the modeling elements of Redo to describe these, such as inside goal definitions, context definitions, resource types, and capability definitions. In addition, each modeling element in Redo is uniquely identifiable. For example, each definition of goals should be uniquely identifiable to enable their association with definitions of resource-driven processes. To enable the identification of different elements of Redo, we present entity identities as follows:

Definition 14 (Entity Identity). Let an entity identity be $id \in ID$. Each such entity identity is a pair:

$$ID \subseteq \Sigma^{+} \times \Sigma^{*}$$

$$id = (namespace, name)$$

Where:

- $\pi_1(id) = namespace$, $namespace \in \Sigma^{+}$ is a globally unique nonempty string representing the namespace of an entity,
- $\pi_2(id) = name, name \in \Sigma^{*}$ is a unique string representing the name of an entity. ♣

A *name* and a *namespace* pair are a unique combination enabling the identification of modeling elements under an organizational definition (Definition 18). Similarly, entity identities can be globally used to identify unique organizational definitions. For example, if an entity identity addresses a human resource, the namespace is a globally unique string specifying the human resource domain in the respective organization. Moreover, the name is a unique string identifying the human resource under that domain, such as the unique organizational identity. Furthermore, multiple modeling elements can share the same *namespace*. To address these different elements sharing the same *namespace*, it is possible to define entity identities without the *name* element. Thus, the *namespace* element of entity identities is also considered a domain identifier. Obviously, the combination of entity definitions and entity identities can be used to describe different modeling elements in a distinct way. For example, business experts will specify a Java developer by combining an entity identity singling the developer out, with the corresponding entity definitions, such as a description of the developer's Java experience. Thus, we present the following identifiable entity definition facilitating the specification of identifiable modeling elements in organizations.

Definition 15 (Identifiable Entity Definition). Let an identifiable entity definition be $ie \in IE$. Each identifiable entity definition is a triple:

$$IE \subseteq ID \times (ID \cup \emptyset) \times \wp(ED)$$

$$ie = (id, id_{sourceModel}, ED_{ie})$$

Where:

- $\pi_1(ie) = id$, $id \in ID$ is an entity identity used to distinguish the represented entity,
- $\pi_2(ie) = id_{sourceModel}$, $id_{sourceModel} \in ID \cup \emptyset$ is an entity identity referring to an entity, which was used to create the entity described by this identifiable entity definition, such that leaving $id_{sourceModel}$ empty implies no existing entity was used to create the represented entity,
- $\pi_3(ie) = ED_{ie}$, $ED_{ie} \subseteq \wp(ED)$ is a set of entity definitions describing the represented entity. ♣

Identifiable entity definitions are used to describe various elements of Redo, such as organizational definitions (Definition 9), resource types (Definition 21), relationship types (Definition 23), capability definitions (Definition 22), resource models (Definition 28), property definitions (Definition 19), operation endpoints (Definition 26), and context definitions (Definition 27). These modeling elements contain an identifiable modeling element resulting in (i) a detailed definition of the modeling elements and (ii) their unique identification using (i) entity definitions (ED_{ie}) and (ii) entity identities (id), respectively. The reason of introducing multiple entity definitions for each identifiable entity definition is possible different (i) representations of a modeling element and (ii) types of information regarding the described element. For example, a software developer can be described in one entity definition with plain text and in another one structured XML data. Moreover, one entity definition can describe skills of the software developer, whereas another one hourly rates for the skills provided by the software developer.

Modeling elements specify change throughout the lifetime of an organization. During these changes, modeling elements can be created based on other modeling elements, such as creating the definition of a goal aimed at fixing a mobile device defect based on the definition of a goal aimed at developing a mobile device due to the common capabilities for both goals. Obviously, retaining the information regarding which modeling elements used for creating which other modeling elements can be useful. For example, if a goal aimed at developing a mobile device is updated, goals created based on this goal may also be updated. Keeping the information regarding the source elements will ease finding modeling elements created based on other modeling elements. Therefore, the element $id_{sourceModel}$ of identifiable entity definitions links modeling elements to their source modeling elements, if applicable. For example, an organization can base a resource-driven process definition representing the development of a product on a resource-driven process definition representing fixing defects in a product because the latter is also part of the former process. Moreover, after adding a new standard diagnostics service to the development process, it will be reasonable to update the resource-driven process definition representing fixing defects. Therefore, $id_{sourceModel}$ can be used to identify the source of a modeling element. Notably, a modeling element represented by $id_{sourceModel}$ and a modeling element created based on $id_{sourceModel}$ may have no more shared attributes eventually contrary to object-oriented inheritance relationships. Consequently, $id_{sourceModel}$ provides a traceability link between a base modeling element and a new modeling element for documentation purposes.

As described in Section 3.2, the approach of resource-driven processes aims at not only modeling business processes in a resource-driven way but also the initializing definitions of resource-driven processes by automatically allocating interrelated resources. Initializing resource-driven processes raises the need for a concept to represent instances of different modeling elements, such as resources, resource-driven processes, and goals. Therefore, we present instance descriptors:

Definition 16 (Instance Descriptor). Let an instance descriptor be $i \in \mathcal{I}$. Each such instance descriptor is a nine-tuple:

$$\mathcal{I} \subseteq IE \times \Sigma^+ \times ID \times (ID \cup \emptyset) \times Q \times \Sigma^* \times \{true, false\} \times \{true, false\} \times [0, 10]$$

$$\begin{aligned} i = (&ie, instanceUri, id_{sourceModel}, id_{parent}, state, \\ &dueDate, propagateSuccess, propagateError, importance) \end{aligned}$$

Where:

- $\pi_1(i) = ie$, $ie \in IE$ is an identifiable entity definition specifying the represented instance,
- $\pi_2(i) = instanceUri, instanceUri \in \Sigma^+$ is a globally unique nonempty string specifying the unified resource locator of the represented instance,
- $\pi_3(i) = id_{sourceModel}, id_{sourceModel} \in ID$ is an entity identity referring to the source model of the represented instance,
- $\pi_4(i) = id_{parent}, id_{parent} \in ID \cup \emptyset$ is an entity identity referring to the instance descriptor specifying the instance causing the initialization of the represented instance, such that leaving id_{parent} empty implies no such instance exists,
- $\pi_5(i) = state, state \in Q$ is a nonempty set of strings specifying the current state of the instance represented by the instance descriptor, such that $Q \subseteq \wp(\Sigma^+)$ is the set of all possible states and $|Q| \in \mathbb{Z}^+$,
- $\pi_6(i) = dueDate, dueDate \in \Sigma^*$ is a string representing the due date of the represented instance,
- $\pi_7(i) = propagateSuccess, propagateSuccess \in \{true, false\}$ is a Boolean value specifying whether a successful state of the instance will be propagated to its parent instance,
- $\pi_8(i) = propagateError, propagateError \in \{true, false\}$ is a Boolean value

specifying whether an erroneous state of the instance will be propagated to its parent instance,

- $\pi_9(i) = importance$, $importance \in [0, 10] \subset \mathbb{N}$ is an integer representing the importance of this instance, such that the value 10 indicates the highest importance. ♣

Instance descriptors represent instances of modeling elements created in the context of resource-driven process executions, such as instances of resources, resource-driven processes, and goals. Naturally, instances of modeling elements can be uniquely identified, such as the unique identification of a resource participating in an instance of a resource-driven process. Therefore, identifiable entity definitions specify instance descriptors uniquely in an organizational definition. Moreover, it is necessary to understand whether two instances described by two instance descriptors actually refer to the same instance. Therefore, we present the string *instanceUri*, which is capable of identifying globally unique instances. For example, two instance descriptors referring to the same Wiki service can have different entity identities. However, they will have the same *instanceUri*, such as the Unified Resource Locator (URL) identifying the Wiki service. To this end, each resource allocated to resource-driven process goals (i.e., resource instance) is represented by an instance descriptor. Furthermore, to identify modeling elements used for creating instances represented by instance descriptors, $id_{sourceModel}$ is used. For example, an instance descriptor representing a resource-driven process will point to the definition of the resource-driven process using $id_{sourceModel}$.

If an instance triggers the initialization of an instance of another modeling element, the instance descriptor of the initialized modeling element refers to the initiator using id_{parent}. For example, during the initialization of a resource-driven process, resources defined in its definition are allocated. Consequently, a resource-driven process triggers the initialization of resources. In this case, id_{parent} of instance descriptors of allocated resources refers to the instance descriptor describing the resource-driven process triggering the allocation. Instance descriptors specify instances of different modeling elements in Redo, such as resource definitions and resource-driven process definitions.

Furthermore, each instance goes through different life cycle states, such as resources going through “pending,” “allocating,” “allocated,” “deallocating,” “failed,” and “deallocated” states during their life cycle. The *state* element of each instance descriptor specifies the current state of an instance.

The state changes can affect the state of the parent instance that initialized an instance. For example, if the allocation of an important resource fails, it will not make much sense to start a resource-driven process that triggers the allocation of the resource. Similarly, accomplishing important goals of a resource-driven process can imply successful completion without completing other goals. For example, achieving a goal of fixing a defect can result in completing a resource-driven process aimed at this goal without achieving a goal of making an impact analysis of the defect. For such state changes affecting the state of parent-instance initiated modeling elements, instance descriptors contain *propagateSuccess* and *propagateError*. To this end, setting *propagateSuccess* to true results in the propagation to its parent of a successful termination of an instance, such as the completed state of a goal, by changing the state of the parent instance to the equivalent successful state. Successful states represent the initialization and finalization of resource-driven processes, completion of goals, allocation of resources, and establishment of relationships. For example, a successful completion of a goal is propagated to its corresponding parent resource-driven process, resulting in a state change in the parent representing the successful finalization of the process. In contrast, setting *propagateError* to true enables the propagation of an erroneous state to a parent instance, such as the termination of a resource-driven process based on an important resource that cannot be allocated for the process. The *propagateSuccess* and *propagateError* elements resemble the *suppressOnFailure* attribute of activities, enabling the suppression of the propagation of a failure in WS-Business Process Execution Language [OAS07b].

Organizations can plan to allocate their resources to resource-driven processes for a certain amount of time. Similarly, goals of resource-driven processes can also have foreseeable deadlines. Thus, instances of different modeling elements, such as goals, resource-driven processes, and resources, can have due dates representing when an instance should be kept active, initialized, and allocated. To specify such a due date of instances, the field *dueDate* exists. Instances can have different degrees of the importance in the scope of different resource-driven processes. For example, the accomplishment of a goal of conducting an impact analysis of a defect in a repairing resource-driven process is very important for a serious defect affecting the functionality of a mobile device. In contrast, in a minor defect, such a goal will be less important. Thus, there should be a means of dynamically specifying the importance of instances of different modeling elements during enactments of resource-driven processes, such as the importance of resources, relationships between resources, goals, and resource-driven processes themselves. For this purpose, each instance descriptor contains an *importance* element similar to the integer priority in WS-HumanTasks [OAS10a]. Stating the *importance* of instances in resource-driven processes results in different use cases, such as non-strict ordering of goals based on their importance and analyzing the strategic importance of certain resource-driven processes, resources, and relationships for an organization.

To specify modeling elements that can be initialized in Redo, initializable entity definitions are used. Formally, initializable entity definitions are defined as follows:

Definition 17 (Initializable Entity Definition). Let an initializable entity definition be $ad \in AD$. Each initializable entity definition is a pair:

$$AD \subseteq IE \times \wp(\mathcal{I})$$

$$ad = (ie, \mathcal{I}_{ad})$$

Where:

- $\pi_1(ad) = ie$, $ie \in IE$ is an identifiable entity definition specifying the represented initializable entity,
- $\pi_2(ad) = \mathcal{I}_{ad}$, $\mathcal{I}_{ad} \subseteq \wp(\mathcal{I})$ is a set of instance descriptors describing instances of the represented initializable entity. ♣

Initializable entity definitions are used to specify modeling elements that are both (i) uniquely identifiable in an organizational definition and (ii) initializable, such as resource-driven process definitions (Definition 29), goal definitions (Definition 28), resource definitions (Definition 24), and relationship definitions (Definition 25). To track different instances of modeling elements, each initializable entity definition is associated with instance descriptors. For example, a resource-driven process definition can be initialized many times, resulting in multiple instance descriptors. Base elements provide fundamentals to create modeling elements of the resource-driven organizational modeling language, which is presented next.

3.3.3 Modeling Elements of Redo

Business experts use different modeling elements to create definitions of different resource-driven processes. Moreover, based on these definitions, business experts create definitions of resource-driven processes. Formally, modeling elements of Redo are defined as follows:

Definition 18 (Modeling Element). Let a modeling element of Redo be $me \in ME$. Each modeling element me is defined as:

$$me \in PD \cup OE \cup \mathcal{S} \cup \mathcal{C} \cup \mathcal{R} \cup SD \cup RD \cup RM \cup CT \cup \mathcal{G} \cup \mathcal{P}$$

Where:

- PD is the set of all property definitions (Definition 19),
- OE is the set of all operation endpoints (Definition 20),
- $\mathcal{S}$ is the set of all resource types (Definition 21),
- $\mathcal{C}$ is the set of all capability definitions (Definition 22),
- $\mathcal{R}$ is the set of all relationship types (Definition 23),
- SD is the set of all resource definitions (Definition 24),
- RD is the set of all relationship definitions (Definition 25),

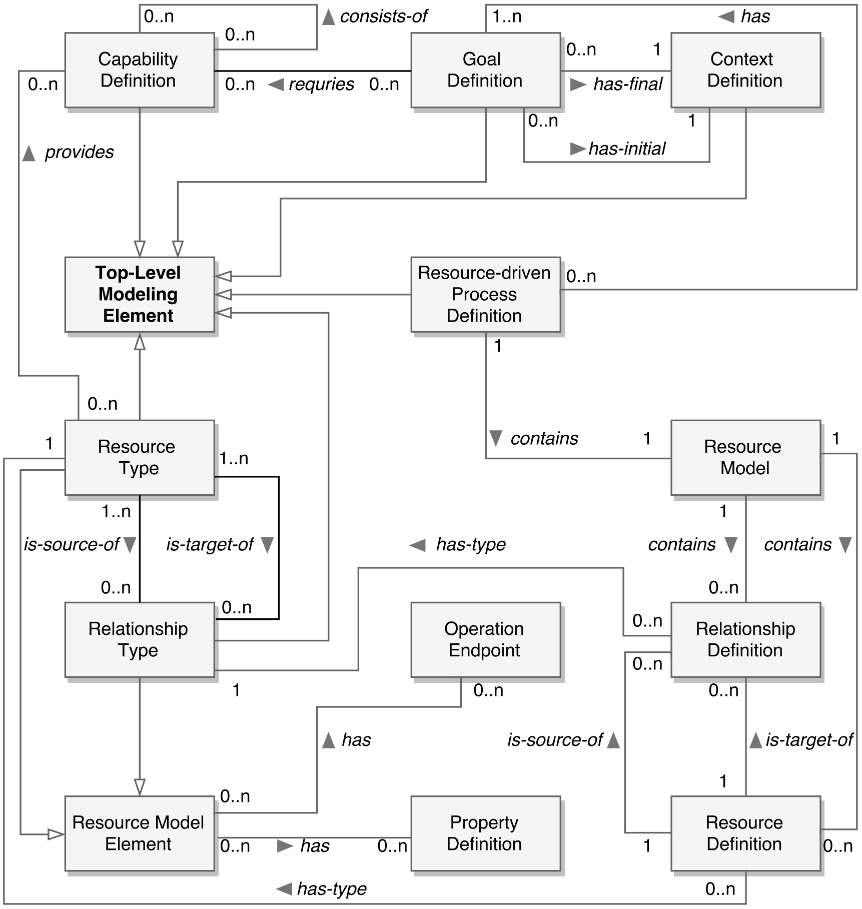


Figure 3.4: Classes and relationships of modeling elements of Redo.

- *RM* is the set of all resource models (Definition 26).
- *CT* is the set of all context definitions (Definition 27).
- $\mathcal{G}$ is the set of all goal definitions (Definition 28),
- $\mathcal{P}$ is the set of all resource-driven process definitions (Definition 29).

♣

In other words, modeling elements are elements of a union set comprising the sets containing different classes illustrated in Figure 3.4. The individual classes of Figure 3.4 will be detailed in the following paragraphs. Different definitions of resource-driven processes represent business processes of organizations in terms of their interrelated resources and goals. Furthermore, different business processes can use the same types of resources, such as two processes using an SQL database. Unfortunately, defining resources and their relationships from scratch during the definition of each resource-driven process can be expensive. Furthermore, business experts may not be aware of all available resources and their relationships, resulting in definitions of resource-driven processes with missing resources. To avoid this shortcoming, business experts can specify different resource types and relationship types in their organizations a priori, such as a resource type representing a database. Consequently, business experts can reuse the available types of resources and relationships during the definition of resource-driven processes instead of specifying each definition of resources from scratch. In addition, the risk of overlooking available interrelated resources will be decreased due to the available types of resources. Naturally, different definitions of resources and relationships created based on the same types can have different properties, such as a database with different admin credentials. Thus, each type contains a set of definitions of properties that can vary between different resource and relationship definitions created based on the resource and relationship types. To specify these varying properties, property definitions are used. Formally, property definitions are defined as follows:

Definition 19 (Property Definition). Let a property definition be $pd \in PD_{od}$ in an organizational definition *od*. Property definitions are a subset of identifiable entity definitions $PD_{od} \subseteq IE_{od}$ available in an organizational definition *od*. ♣

Property definitions specify the structure of properties varying for each resource and relationship definition. To this end, customizable properties are the properties set by business experts creating definitions of resources and their relationships when modeling resource-driven processes. For example, an XML schema specifying the credentials of a database can contain a complex type with an element for the user name and an element for the password. Such a schema will be stored under the *definitionContent* of an entity definition with its corresponding *type* element identifying the XML schema. Moreover, a property definition can contain the entity definition of credentials for specifying customizable properties of a database resource.

Upon the initialization of resource-driven processes, interrelated resources are automatically allocated. Enabling the automated allocation of resources requires the invocation of operations capable of completing the allocation. Furthermore, during the execution of resource-driven processes, operations provided by resources can be executed. For example, a human resource can provide an operation for interaction. Similarly, a resource representing a simulation process can offer an operation to retrieve intermediary results. Finally, establishing relationships among resources can require the execution of a certain business logic by executing an operation, such as an operation adding a new manager to MediaWiki to establish a "managing" relationship. To represent such operations for various types of resources and relationships, we use operation endpoints.

Definition 20 (Operation Endpoint). Let an operation endpoint be $oe \in OE_{od}$ in an organizational definition od. Each such operation endpoint is a quintuple:

$$OE_{od} \subseteq IE_{od} \times ID \times \Sigma^{+} \times \Sigma^{+} \times \wp(ID)$$

$$oe = (ie, id_{operationType}, endPointType, endPointUri, ID_{oe})$$

Where:

- $\pi_1(oe) = ie$, $ie \in IE_{od}$ is an identifiable entity definition specifying the represented operation,
- $\pi_2(oe) = id_{operationType}$, $id_{operationType} \in ID$ is a globally unique entity identity referring to the type of the represented operation,
- $\pi_3(oe) = endPointType, endPointType \in \Sigma^{+}$ is a globally unique non-empty string referring to the type of this endpoint,

- $\pi_4(oe) = endPointUri$, $endPointUri \in \Sigma^+$ is a globally unique nonempty string specifying the unified resource locator of the operation endpoint,
- $\pi_5(oe) = ID_{oe}$, $ID_{oe} \subseteq \wp(ID)$ is an ordered set of entity identities (the actual order is not important) referring to operation types that are needed to execute the operation specified by the operation endpoint. ♣

Operation endpoints enable the execution of operations on resources and relationships, such as an allocation operation of a Wiki service and an establishment operation for creating a "managing" relationship between a Wiki service and a human resource. Because operation endpoints can be passed around individually, it is important to enable their identification in an organizational definition. Therefore, operation endpoints are specified using identifiable entity definitions with entity identities, enabling their identification in an organizational definition. For example, an operation endpoint can represent the allocation operation of a Wiki service with an entity identity containing the name "allocate" and the namespace of the Wiki service. Consequently, an entity identity of an operation endpoint is used to identify the respective endpoint within an organizational definition. Thus, the same entity identity can represent another modeling element in another organizational definition.

Using uniquely identifiable entity identities of operation endpoints, it is possible to execute allocation operations upon the initialization of resource-driven processes declaratively. For example, allocating a Wiki service will require executing the operation represented by an entity identity containing the name "allocate" and the namespace of the Wiki service. Unfortunately, automating the allocation of resources based on nonglobally unique entity identities of operation endpoints will require changing the business logic for the automation in another organization, as entity identities of the allocation operations endpoint may change. This will decrease the portability of definitions containing operation endpoints between different organizations. To avoid such portability problems, operation endpoints provide a globally unique $id_{operationType}$ capable of identifying operations that are independent of organizational definitions containing the operation endpoints, such as identifying the “allocation” operation based on the value of $id_{operationType}$ elements of different operation endpoints.

To execute a selected operation endpoint, a service consumer initially uses the information in *endPointType*. To this end, we assume that the service consumer of an operation endpoint is capable of (i) understanding *endPointType* and (ii) calling the operation endpoint based on the information contained in $id_{operationType}$ and *endPointType*. For example, a service consumer can understand that an operation endpoint of an allocation operation is a web service defined using WSDL based on the contents of *endPointType*. Furthermore, the service consumer can resolve the interaction protocol, input parameters, and output parameters of the operation endpoint by checking a registry for the operation type $id_{operationType}$. The consumer uses *endPointUri* to call the operation, such as a web service invocation over HTTP using the standard parameters needed for allocating the resource. Thus, further information regarding the invocation of a service, such as input and output parameters of operations, is out of the scope of operation endpoints and is assumed to be defined somewhere else.

Executing an operation represented by an operation endpoint on its supported resource and relationship types (Definition 21 and Definition 23) can necessitate the existence of other operation endpoints available on the resource and relationship types. For example, an operation endpoint supporting an “allocation” operation on the IT resources can require another operation endpoint providing CSAR files, as it uses OpenTOSCA to allocate the respective IT resource. Therefore, each operation endpoint specifies certain required operation types for executing an operation endpoint (ID_{oe}). If a required operation type is not available in a resource or relationship type, the corresponding operation endpoint cannot be executed. For example, the allocation operation using the OpenTOSCA container for the allocation cannot be executed if an operation endpoint providing a CSAR file of the corresponding resource type is not available. In summary, operation endpoints enable a means of executing operations of resources and relationships, such as an allocation operation specified for a certain type of resource.

To describe different types of resources Redo, business experts use resource types. Formally, each resource type is defined as follows:

Definition 21 (Resource Type). Let a resource type be $s \in \mathcal{S}_{od}$ in an organizational definition *od*. Each resource type is a quintuple:

$$\mathcal{S}_{od} \subseteq IE_{od} \times \{true, false\} \times \wp(PD_{od}) \times \wp(ID_{od}) \times \wp(OE_{od})$$

$$s = (ie, unique, PD_s, ID_s, OE_{sr})$$

Where:

- $\pi_1(s) = ie$, $ie \in IE_{od}$ is an identifiable entity definition specifying the represented resource type,
- $\pi_2(s) = unique$, $unique \in \{true, false\}$ is a Boolean value specifying whether the represented resource is unique or not,
- $\pi_3(s) = PD_s$, $PD_s \subseteq \wp(PD_{od})$ is a set of property definitions specifying the customizable properties of the represented resource type,
- $\pi_4(s) = ID_s$, $ID_s \subseteq \wp(ID_{od})$ is a set of entity identities referring to capability definitions provided by the presented resource type,
- $\pi_5(s) = OE_{sr}$, $OE_{sr} \subseteq \wp(OE_{od})$ is a set of operation endpoints providing operations for the defined resource. ♣

Resource types specify different types of resources available in an organization, such as a certain human resource (e.g., Oliver Kopp), an SQL database, or a simulation machine. Moreover, resource types can represent (i) multiple or (ii) single resources, such as (i) roles and an SQL database representing multiple people and databases or (ii) specific individuals and databases containing certain data. The reason for associating single resources is the possible positive effect of particular individuals and information available in these resources on business processes. For example, to resolve a defect of a mobile device, a Wiki service containing the information relevant to the mobile device will be allocated, but not just any Wiki service.

To facilitate their unique identification in an organizational definition, each resource type is specified by an identifiable entity definition. Furthermore, to identify particular resources, such as particular individuals and databases containing certain data, we provide a Boolean flag called *unique* in resource types. We distinguish these resources from other more generic resources, as they are typically managed with more care. For example, a database containing sensitive data is not deallocated without storing the data efficiently. The identifiable entity definition of resource types contains the information to identify the type of unique resource in an organizational definition. Moreover, different entity definitions of a resource type specify different properties of the resource. For example, an entity definition can specify skills of a resource type of software developer. Customizable properties of a resource type are described using the property definition elements PD_s, such as an XML schema describing the required credentials of a resource type representing a database.

Furthermore, organizational resources provide capabilities to the organization as explained in Section 3.2. Therefore, resource types refer to capabilities provided by a certain type of resource using their entity identities, such as a software development capability provided by a certain person. Each capability is a separate construct in Redo. Formally, capabilities are defined as follows:

Definition 22 (Capability Definition). Let a capability definition be $c \in \mathcal{C}_{od}$ in an organizational definition od. Each capability definition is a triple:

$$\mathcal{C}_{od} \subseteq IE_{od} \times \wp(ID_{od}) \times \wp(ID_{od})$$

$$c = (ie, ID_s, ID_c)$$

Where:

- $\pi_1(c) = ie$, $ie \in IE_{od}$ is an identifiable entity definition specifying the represented capability,
- $\pi_2(c) = ID_s$, $ID_s \subseteq \wp(ID_{od})$ is a set of entity identities referring to re-

source types providing the represented capability,

- $\pi_3(c) = ID_c$, $ID_c \subseteq \wp(ID_{od})$ is a set of entity identities referring to capability definitions composing the represented capability. ♣

Capability definitions specify abilities of organizations to conduct productive activities, such as capabilities for Java development or English-speaking. To identify these capabilities uniquely, they are specified using identifiable entity definitions. Moreover, capability definitions refer to different resource types providing the capabilities. Interestingly, complex goals require more complex capabilities covering multiple functions available in organizations, which are cross-functional capabilities. For example, a development project requires a capability for managing projects comprising multiple functional and other cross-functional capabilities, such as capabilities for managing finances and deadlines. Therefore, capability definitions contain entity identities referring to other capability definitions comprising the specified capability definition. During the execution of resource-driven processes, organizations allocate a group of resources to build such cross-functional capabilities.

To increase the performance, security, privacy, and functionality of allocated resources, relationships are used (Section 3.1.2). Similar to resources, many relationships are shared among different business processes. Therefore, specifying different types of available relationships can increase the reuse and decrease the effort to describe the same relationship repeatedly for each different definition of resource-driven processes containing the relationship. For example, a human resource can have a "managing" relationship with a Wiki service in several resource-driven processes. Obviously, specifying a reusable relationship type representing the "managing" relationship will decrease the effort of repeatedly defining the relationship from scratch. To represent different types of relationships in organizations, we introduce the concept of relationship types:

Definition 23 (Relationship Type). Let a relationship type be $r \in \mathcal{R}_{od}$ in an organizational definition od. Each relationship type is a quintuple:

$$\mathcal{R}_{od} \subseteq IE_{od} \times \wp(PD_{od}) \times \wp(ID_{od}) \times \wp(ID_{od}) \times \wp(OE_{od})$$

$$r = (ie, PD_r, ID_{sSource}, ID_{sTarget}, OE_{sr})$$

Where:

- $\pi_1(r) = ie$, $ie \in IE_{od}$ is an identifiable entity definition specifying the represented relationship type,
- $\pi_2(r) = PD_r$, $PD_r \subseteq \wp(PD_{od})$ is a set of property definitions specifying the customizable properties of the relationship type,
- $\pi_3(r) = ID_{sSource}$, $ID_{sSource} \subseteq \wp(ID_{od})$ is a set of entity identities referring to possible source resource types of the presented relationship type, such that $1 \leq |ID_{sSource}|$,
- $\pi_4(r) = ID_{sTarget}$, $ID_{sTarget} \subseteq \wp(ID_{od})$ is a set of entity identities referring to possible target resource types of the presented relationship type, such that $1 \leq |ID_{sTarget}|$,
- $\pi_5(r) = OE_{sr}$, $OE_{sr} \subseteq \wp(OE_{od})$ is a set of operation endpoints providing operations for the relationship type defined. ♣

To specify different types of relationships among organizational resources, relationship types are used. Relationship types can represent historical or desired facts between two resources. For example, a relationship type can represent a “has-worked-with” relationship between a human resource and a database containing a specific dataset, implying the existence of a historical fact of a past work experience between the connected resources. In contrast, a “leading” relationship between two human resources will indicate a desired fact that should be satisfied after starting the enactment of a resource-driven process. To this end, relationships representing desired facts will contain typically operation endpoints capable of establishing the desired facts. For example, a “managing” relationship between a project owner and a Wiki service will require executing a business logic adding an administrator to the corresponding Wiki service.

To facilitate their unique identification in an organizational definition, each relationship type is specified by an identifiable entity definition. Furthermore, each relationship among resources can contain certain customizable properties depending on the resource-driven processes used. For example, a “using” relationship between a human actor and a Git service can require an admin password to establish such a relationship. The property definition elements of a relationship type are used to prescribe such properties of the resource relationships.

Each relationship in an organization is between different types of resources. For example, an “is-a-friend-with” relationship can be specified only among human resources. To specify the types of resources that can be connected by relationship types, relationship types contain a set of source and target entity identities referring to these resource types. In certain cases, a relationship can connect every resource in a certain domain identified by a namespace, such as an “is-a-friend” relationship possibly connecting every human resource in an organization. Moreover, defining entity identities for each possible target and source resource type of relationship type can be unfeasible due to the effort spent on specifying many entity identities. Instead, to address all resource types specified with a common namespace, the name field of an entity identity referring to the target and source resource types can be left empty.

Establishing relationships upon the initialization of resource-driven processes can require the execution of a certain business logic, such as invoking an operation for establishing a relationship between a Wiki service and a human resource represented by an operation endpoint providing the operation. Resource types and relationship types are reusable elements used in definitions of resource-driven processes to create definitions of resources and relationships. To specify resources of business processes based on resource types, we introduce resource definitions. Formally, resource definitions are defined as follows:

Definition 24 (Resource Definition). Let a resource definition be $sd \in SD_{od}$ in an organizational definition od. Each resource definition is a quadruple:

$$SD_{od} \subseteq AD_{od} \times ID_{od} \times \{coupled, uncoupled, onDemand\} \times \wp(\Sigma^*)$$

$$sd = (ad, id_{sType}, allocationBehavior, \mathcal{A}_{sd})$$

Where:

- $\pi_1(sd) = ad$, $ad \in AD_{od}$ is an initializable entity definition specifying the represented resource definition,
- $\pi_2(sd) = id_{sType}$, $id_{sType} \in ID_{od}$ is an entity identity referring to the resource type of the represented resource definition,
- $\pi_3(sd) = allocationBehavior$, $allocationBehavior \in \{coupled, uncoupled, onDemand\}$ is a value specifying the allocation behavior of the defined resource, such that:
  - the default value *coupled* results in an immediate allocation of the resource upon the initialization of a resource-driven process, terminating the resource-driven process in case of a failure during the allocation,
  - the value *uncoupled* results in an immediate allocation of the resource upon initializing a resource-driven process without terminating the resource-driven process in the case of a failure during the allocation,
  - the value *onDemand* results in an on-demand allocation at runtime,
- $\pi_4(sd) = \mathcal{A}_{sd}$, $\mathcal{A}_{sd} \subseteq \wp(\Sigma^*)$ is a set of customizable properties created based on the property definitions of the resource type. ♣

Resource definitions describe resources of resource-driven processes based on resource types created previously, such as a resource definition of a MediaWiki service used in a software development project based on a generic

MediaWiki resource type. Moreover, upon initializing definitions of resource-driven processes, described resources can be allocated. Thus, resource definitions are specified using initializable entity definitions with instance descriptors representing different instances of resource definitions. Interestingly, certain resources are used only if they are needed and will not be required otherwise. Allocating such resources upon initializing resource-driven processes will result in a waste of resources that are actually not used. To avoid this, we present the *allocationBehavior* value in resource definitions. Setting *allocationBehavior* to the value *onDemand* will enable the allocation of resources based on the requirements of actual enactments in an ad hoc fashion after initializing resource-driven processes instead of directly allocating resources. In contrast, the values *uncoupled* and *coupled* will result in immediately allocating resources after initializing resource-driven processes. Moreover, the main difference between these two values is the failure behavior during allocation. When the allocation of a resource with the *allocationBehavior* set to *uncoupled,* the initialization of the parent resource-driven process proceeds, even if the allocation of the resource fails. Thus, the *propagateError* of the created instance descriptor for the resource is set to false (Definition 16). Alternatively, if the allocation of a resource specified with *coupled* fails, the initialization of the resource-driven process is terminated as well, and its already allocated resources are released. Consequently, *propagateError* of the respective instance descriptor, representing the new resource, is created with the value of *coupled* set to true.

Each resource type referred to by the entity identity id_{sType} in a resource definition represents a basis for resource definitions. Moreover, it is possible to tailor resource definitions to different requirements for different resource-driven processes using available customization points provided in property definitions of the resource type. Each property in the set of properties $\mathcal{A}_{sd}$ is a string. Moreover, the structure of the string is specified in a property definition of the resource type referred to by the resource definition. For example, if a property definition describes the admin credentials of a database resource, the respective property will contain a string specifying the admin credentials following the structure in the respective property definitions.

Similar to the resource definitions, there are relationship definitions connecting these in resource-driven processes:

Definition 25 (Relationship Definition). Let a relationship definition be $rd \in RD_{od}$ in an organizational definition od. Each such relationship definition is a six-tuple:

$$RD_{od} \subseteq AD_{od} \times ID_{od} \times ID_{od} \times ID_{od} \times \{true, false\} \times \wp(\Sigma^*)$$

$$rd = (ad, id_{rType}, id_{sdSource}, id_{sdTarget}, mandatory, \mathcal{A}_{rd})$$

Where:

- $\pi_1(rd) = ad$, $ad \in AD_{od}$ is an initializable entity definition specifying the represented relationship,
- $\pi_2(rd) = id_{rType}$, $id_{rType} \in ID_{od}$ is an entity identity referring to the relationship type of the represented relationship,
- $\pi_3(rd) = id_{sdSource}$, $id_{sdSource} \in ID_{od}$ is an entity identity referring to the source resource definition of the represented relationship, such that the resource type of the source resource definition is in the list of possible source resource types of the relationship type,
- $\pi_4(rd) = id_{sdTarget}$, $id_{sdTarget} \in ID_{od}$ is an entity identity referring to the target resource definition of the represented relationship, such that the resource type of the target resource definition is in the list of possible target resource types of the relationship type,
- $\pi_5(rd) = mandatory$, $mandatory \in \{true, false\}$ is a Boolean value specifying the necessity of the relationship in a resource-driven process, such that true is the default value, meaning the relationship must be established,
- $\pi_6(rd) = \mathcal{A}_{rd}$, $\mathcal{A}_{rd} \subseteq \wp(\Sigma^*)$ is a set of customizable properties created using property definitions of the relationship type. ♣

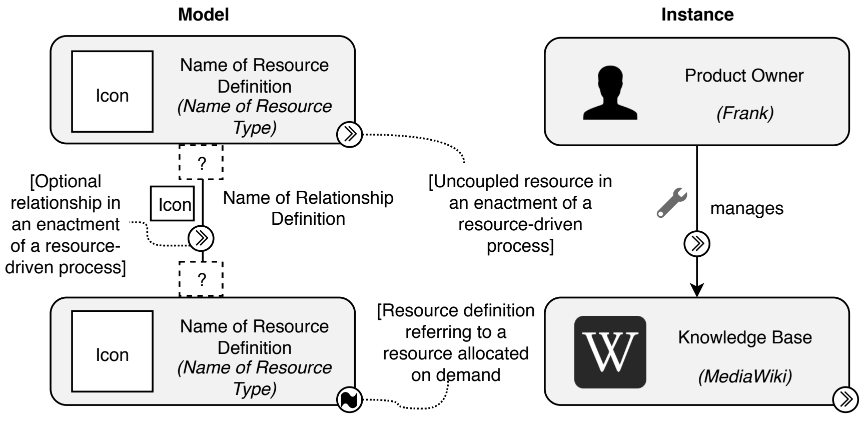

Model
Instance
Icon
Name of Resource Definition
(Name of Resource Type)
Product Owner
(Frank)
?
[Optional relationship in an enactment of a resource-driven process]
Icon
Name of Relationship Definition
[Uncoupled resource in an enactment of a resource-driven process]
manages
?
Icon
Name of Resource Definition
(Name of Resource Type)
[Resource definition referring to a resource allocated on demand]
Knowledge Base
(MediaWiki)

Figure 3.5: Illustration of resource models based on graphical notations of Breitenbücher et al. [Bre16; BBK+12].

Relationship definitions present different relationships observed between resources of resource-driven processes during their enactments, such as a “managing” relationship or a “has-worked-together” relationship. Furthermore, business experts specify relationship definitions based on certain relationship types identified by the entity identity id_{rType}. As relationships between resources can be automatically established in a resource-driven process, relationships are specified using initializable entity definitions holding the information on the instances of documented relationships of resource-driven processes. Because of errors, it may not be possible to establish every relationship specified between resources upon the initialization of resource-driven processes. Furthermore, certain relationships can be required for performance purposes, however, a resource-driven process can be executed even if they cannot be established. To enable the specification of such relationships, the *mandatory* field of relationship definitions is used. Setting *mandatory* of a relationship definition to `true` will stop the initialization and terminate the respective resource-driven process containing the relationship upon a failure during its establishment. Each relationship definition connects two resource definitions identified by $id_{sdSource}$ and $id_{sdTarget}$. Moreover, it is possible to customize relationship definitions for different requirements of various resource-driven processes using available customization points in property definitions of the relationship type. Property definitions specified in the referred relationship type provide meta-data about the properties of relationship definitions. Properties of relationship definitions follow the structure of these property definitions. Resource definitions and relationship definitions build a graph called a resource model, which is defined as follows:

Definition 26 (Resource Model). Let a resource model be $rm \in RM_{od}$ in an organizational definition od. Each resource model is a possibly cyclic directed graph defined as follows:

$$RM \subseteq \wp(SD_{od}) \times \wp(RD_{od})$$

$$rm = (SD_{rm}, RD_{rm})$$

Where:

- $\pi_1(rm) = SD_{rm}$, $SD_{rm} \subseteq \wp(SD_{od})$ are resource definitions representing vertices of a resource model,
- $\pi_2(rm) = RD_{rm}$, $RD_{rm} \subseteq \wp(RD_{od})$ are relationship definitions representing edges of a resource model.

Resource models can contain isolated vertices and multiple edges between vertices. ♣

Each resource model has at least one resource definition. Figure 3.5 presents a graphical notation for modeling resource models based on the graphical notation proposed by Breitenbücher et al. [Bre16; BBK+12]. In our notation, each rounded rectangle stands for a resource definition of a resource model. Furthermore, edges between resource definitions represent relationship definitions. Each resource definition and relationship definition provides an icon for increasing their comprehension. In resource models, we do not show types of relationship definitions, as these do not increase the expressiveness of resource models in our case. The circle on the right bottom corner of resource definitions in Figure 3.5 provides information regarding the allocation behavior of the respective resource definition (*allocationBehavior*). The graphical notations representing the allocation behavior are shown at the right bottom corner only if (i) the allocation of a resource is optional (*uncoupled*) and (ii) the resource will be allocated on-demand during the enactment of a resource-driven process (*onDemand*). Otherwise, a resource will be allocated immediately (*coupled*), which is not shown. Similarly, a relationship definition can be specified as optional. In such a case, the same icon is shown on the relationship definition as in the case of resource definitions. Please note that all relationships between an *uncoupled* resource and a resource must be, as well, optional because of consistency rules.

Organizations react to different situations differently. For example, upon the presence of a defect in a mobile device, the corresponding organization can execute a business process aimed at fixing the defect. Moreover, it will be beneficial for organizations to document these situations, so that situations can be identified and best practices for the situations can be created, such as best practices for handling the defect in a mobile device. In Redo, different situations are described using context definitions:

Definition 27 (Context Definition). Let a context definition be $ct \in CT_{od}$ in an organizational definition od. Context definitions compose a subset of identifiable entity definitions $CT_{od} \subseteq IE_{od}$ available in an organizational definition od. ♣

Context definitions specify particular situations in organizations, such as the presence of a defect. Therefore, context definitions are specified using an identifiable entity definition. As each identifiable entity definition contains multiple entity definitions, entity definitions can be used to describe a context differently, such as in human-readable format or machine-processable format. Thus, humans can use a human-readable format to understand the definition of a context. Moreover, web services can automatically recognize the existence of a context defined using a machine-processable format.

Organizations aim at staying in certain contexts or at reaching those contexts representing better states. For example, a production company will try to stay in a context representing a functioning mobile device without any surfaced defects produced by the company. In contrast, in the presence of a context representing a malfunctioning device with defects, the production company will try to reach the context representing a functioning device. To stay in a certain desired context and to reach a context specifying a better state, organizations achieve goals. Formally, each goal is specified as follows:

Definition 28 (Goal Definition). Let a goal definition be $g \in \mathcal{G}_{od}$ in an organizational definition *od*. Each goal definition is a quadruple:

$$\mathcal{G} \subseteq AD \times ID_{od} \times ID_{od} \times \wp(ID_{od})$$

$$g = (ad, id_{ctInitial}, id_{ctFinal}, ID_c)$$

Where:

- $\pi_1(g) = ad,\ ad \in AD$ is an initializable entity definition specifying the represented goal,
- $\pi_2(g) = id_{ctInitial}, id_{ctInitial} \in ID_{od}$ is an entity identity referring to a context definition specifying a situation in which a certain goal needs to be achieved,
- $\pi_3(g) = id_{ctFinal}, id_{ctFinal} \in ID_{od}$ is an entity identity referring to a context definition specifying situation, in which achieving the goal has been concluded,
- $\pi_4(g) = ID_c, ID_c \subseteq \wp(ID_{od})$ is a set of entity identities referring to a set of capability definitions to achieve the goal. ♣

Goal definitions describe goals of organizations, which are achieved by executing business processes in organizations, such as a goal to fix a defect in a mobile device. Moreover, different business processes can have the same goals. Consequently, in each business process, an instance of a goal specific to the instance of the business process itself exists. Therefore, goal definitions are specified using initializable entity definitions containing different

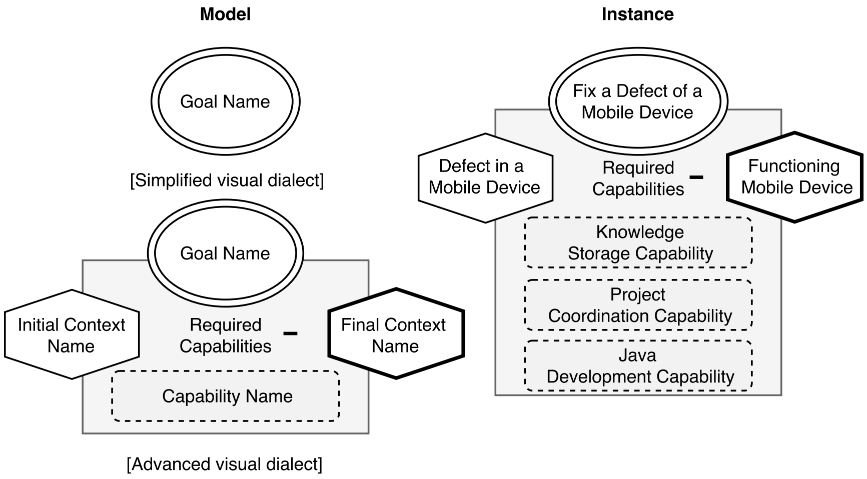

Model
Instance
Goal Name
[Simplified visual dialect]
Goal Name
Initial Context Name
Required Capabilities
Final Context Name
Capability Name
[Advanced visual dialect]
Fix a Defect of a Mobile Device
Defect in a Mobile Device
Required Capabilities
Functioning Mobile Device
Knowledge Storage Capability
Project Coordination Capability
Java Development Capability

Figure 3.6: Illustration of goal definitions.

instances of the goals described and reference their initial and final context definitions using their entity identities. Initial context definitions represent the conditions, in which a goal must be accomplished in organizations, such as the presence of a defect in a mobile device. Similarly, final context definitions represent conditions in which the accomplishment of a goal is over, such as a functioning mobile device. Furthermore, goal definitions point to a set of capabilities for accomplishing goals.

Figure 3.6 presents a graphical notation for representing goal definitions of the language. Each goal definition is represented using two elliptical shapes. Furthermore, depending on the modeling environment, we present a simple visual dialect and advanced visual dialect, as recommended by Moody [Moo09]. The advanced visual dialect of a goal presents the name of the initial and final context and capabilities required to achieve the goal. Thus, each graphical representation of a goal definition provides an overview of the considered goal. Please note that for the sake of clarity, explicit relationships between goals, such as “contradicting” relationships and decomposition relationships are not considered in Redo (Section 2.2). Supporting such relationships between goals can be achieved with an additional attribute in each goal definition. Achieving the goals of an organization requires the execution of business processes. Furthermore, the goals provide a direction for actors of business processes toward the desired outcomes. Therefore resource-driven processes refer to the target goals. Formally, each resource-driven process definition is specified as follows:

Definition 29 (Resource-driven Process Definition). Let a resource-driven process definition be $p \in \mathcal{P}_{od}$ in an organizational definition od. Each resource-driven process definition is a triple specified as follows:

$$\mathcal{P} \subseteq AD_{od} \times RM_{od} \times \wp(ID_{od})$$

$$p = (ad, rm, ID_g)$$

Where:

- $\pi_1(p) = ad, ad \in AD_{od}$ is an initializable entity definition specifying the represented resource-driven process,
- $\pi_2(p) = rm,\ rm \in RM_{od}$ is a resource model specifying resources and their relationships of the resource-driven process,
- $\pi_3(p) = ID_g,\ ID_g \subseteq \wp(ID_{od})$ is a set of entity identities referring to goal definitions targeted by the resource-driven process. ♣

Organizations use resource-driven process definitions to document recurring business processes in a resource-driven way. Furthermore, resource-driven process definitions can be initialized, resulting in the allocation of interrelated resources as specified in resource models *rm* of resource-driven process definitions. Therefore, initializable entity definitions specify resource-driven process definitions. Each resource-driven process definition specifies its objective by referring to a set of goal definitions, as shown in Figure 3.7. The graphical representation of resource-driven processes is based on the graphical representation of resource models and goal definitions presented

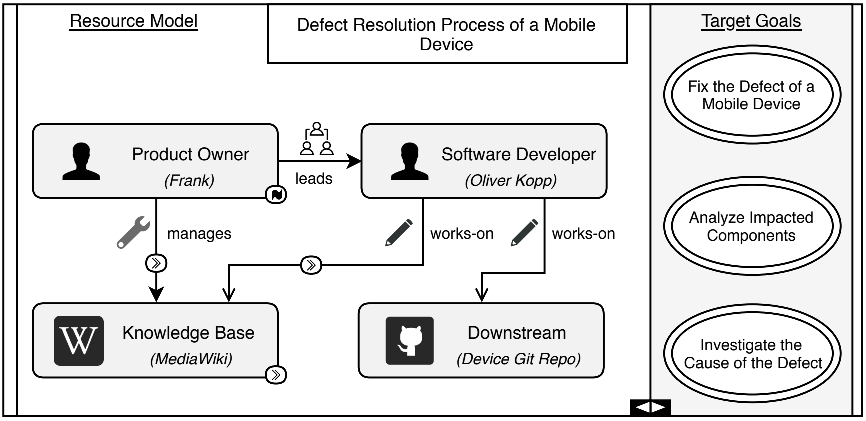


Figure 3.7: Illustration of resource-driven process definitions.

in Figure 3.5 and Figure 3.6, respectively. More specifically, the graphical notation of resource-driven process definitions presents a process pool containing a resource model on the left-hand side and target goals of the corresponding resource-driven process definition on the right-hand side. The separator between resource models and target goals can be adjusted to adapt to the sizes of the different corresponding areas. Allocated interrelated resources of resource-driven processes will work toward these goals to complete the processes successfully. Moreover, the association of goals with capabilities in goal definitions provides implicit semantics for the task distribution of allocated human resources during the enactment of resource-driven processes. Instance descriptors of goal definitions contain the attribute *importance* to prioritize the goals during enactments of resource-driven processes. Consequently, certain implicit relationships between goals exist. Next, we provide an evaluation of the resource-driven approach based on the requirements presented previously in Section 3.1.

Table 3.1: An overview of the evaluation of the resource-driven processes with different approaches.

Approaches for Modeling and Executing Business Processes	**Base Assumption of Unstructured Business Processes (Rq_1)**	**Resource Relationship Definition (Rq_2)**	**Resource Visibility Definition (Rq_3)**	**Support for Dynamically Changing Resources (Rq_4)**
Resource-driven Processes	+	+	+	+
Activity-oriented Approaches	-	∼	∼	∼
Constraint-based Approaches	-	∼	∼	∼
Content-oriented Approaches	+	∼	∼	+

3.4 Evaluation of the Resource-driven Processes Using Requirements

In this section, we evaluate different business process modeling approaches, such as resource-driven processes, activity-oriented approaches, constraint-based approaches, and content-oriented approaches (Section 2.3), using the requirements presented previously in Section 3.1. A summary of the following evaluation is presented in Table 3.1, where +, ∼, and - denote (i) meeting a requirement, (ii) partially meeting a requirement, and (iii) failing to meet a requirement, respectively. Please note that we discuss only the relevant approaches addressing our presented requirements (Section 3.1). For example, we discuss the activity-oriented approach Caramba [Dus04; Dus05] in the context of the Base Assumption of Unstructured Business Processes (Rq_1) because this approach presents relevant concepts addressing the corresponding requirement. Thus, the approaches described in Section 2.3 but not mentioned next do not change the results of our evaluation illustrated in Table 3.1.

3.4.1 Base Assumption of Unstructured Business Processes (Rq_1)

Rq_1 requires an approach for supporting actors of informal processes and reproducing their desired outcomes to be able to work in the case of unstructured business processes (Section 3.1.1). Resource-driven processes rely on automatically allocated interrelated resources to support actors of informal processes and reproduce their desired outcomes. The support for actors is provided by (i) automated allocation of resources, (ii) documenting resources required in enactments of business processes, and (iii) resources themselves. Furthermore, the reproducibility of desired outcomes of informal processes is ensured by allocating interrelated resources capable of generating the business logic needed for reaching the goals of informal processes in an automated fashion. Consequently, resource-driven processes provide support for actors of informal processes and enable the means of reproducing desired outcomes even if no predictable and repeated interrelated activities between different executions of the same informal process exist. Notably, business experts can use organizational resources capable of coordinating actors to increase support in terms of coordination, such as project management tools and activity-oriented business process models.

In contrast, activity-oriented business process modeling and execution approaches (Section 2.3.1) rely on activities for modeling business process models. Consequently, they fail to support unstructured business processes and fail Rq_1, as interrelated activities of unstructured business processes are neither predictable at modeling time nor repeated between different enactments of a business process. Approaches allowing ad hoc changes on business process models, such as Caramba [Dus04; Dus05], can help capture activity structures of business processes at runtime. However, such approaches are unsuitable for reproducing the desired outcomes of unstructured business processes, as captured business process models will be unusable in future enactments of the business process [Dus04; Dus05; Son16b; DR09; SK10; OAS10a]. Using constraint-based approaches for modeling and executing business processes (Section 2.3.2), possible activities at runtime are limited by a set of specified constraints. Consequently, constraint-based approaches relax the strict order of activities as in the case of activity-oriented approaches using constraints. However, constraint-based approaches still require the definition of activities and their constraints. Consequently, constraint-based approaches fail to meet the Rq_1.

Content-oriented approaches focus on information resources, such as business artifacts, and their state changes for representing advancements in business processes (Section 2.3.3). These approaches are more suitable for modeling unstructured business processes, as they typically do not (only) rely on modeling of activities of business processes. Representing the content, that is, information resources, business processes enable support for human actors allocated for business processes. Thus, content-oriented approaches meet the requirement Rq_1.

3.4.2 Resource Relationship Definition (Rq_2)

Defining relationships among resources can increase performance, security, privacy, and functionality of informal processes, as explained in Section 3.1.2. Therefore, resource-driven processes contain the concept of resource relationships. In resource-driven processes, types of relationships between resources are not restricted. More specifically, relationships can represent (i) historical facts and (ii) desired facts between two organizational resources at runtime, such as (i) the fact that two human resources “have-worked-together” in the past and (ii) a human resource manages a MediaWiki resource, respectively. Consequently, resource-driven processes meet requirement Rq_2.

Relationship support provided by the approaches described in Section 2.3 is rather limited. RALTeam enables the assignment of teams to activities in activity-oriented business process models [CRMR15]. However, the relationships between team members are not addressed and, thus, RALTeam fails to meet Rq_2. Furthermore, a set of activity-oriented approaches addresses relationships between human resources during the allocation of resources for activities, such as facilitating the execution of an approval activity by the manager of an actor conducting a vacation request [RtHEvdA04]. Therefore, the logical people groups in the WS-HumanTask specification can be used to specify certain relationships between actors of a business process [OAS10a]. Thus, the relationship support provided by the WS-HumanTask specification is restricted by allowing certain relationships involving human actors only. For example, it is impossible to define a “managing” relationship between ad hoc attachments and human actors. Thus, the WS-HumanTask specification fulfills Rq_2 only partially. Caramba presents a more extended support for relationships in comparison with the WS-HumanTask specification by providing relationships among all Caramba objects, such as human resources, their skills, and database tables [Dus04; Dus05]. However, the provided relationship support is limited to semantic relationships and does not facilitate the automated establishment of relationships as stated in Rq_2. Because of the limited relationship support provided by Caramba and WS-HumanTask, we categorize activity-oriented approaches as partially satisfying requirement Rq_2. Similarly, constraint-based approaches (Section 2.3.2) do not introduce additional concepts regarding modeling and establishment of relationships between resources. Moreover, these approaches allow relationships involving human actors and, thereby, meet Rq_2 only partially.

Content-oriented approaches (Section 2.3.3) generally do not concern relationships among resources participating in business processes. More specifically, the content-oriented approach by Liptchinsky et al. [LKTD12] introduced relationships among information resources and human resources in the context of state transitions of artifacts. For example, if the implementation of a software feature is in a pending state, a transition to the next state will depend on the presence of an employee related to a developer who has already made a contribution to the implementation [LKTD12]. Such relationships describe conditions for making state transitions using relationships among resources. Thus, they are not designed, for example, to restrict access to certain resources. Consequently, the approach provided by Liptchinsky et al. [LKTD12] partially satisfies Rq_2. Because of the limited relationship support provided by their approach, we categorize content-oriented approaches as partially meeting Rq_2.

3.4.3 Resource Visibility Definition (Rq_3)

Resource-driven processes ensure the visibility of resources having an impact on informal processes by introducing a generic resource concept. More specifically, resource-driven processes address the inclusion of human, IT, information, and physical resources in resource-driven process definitions. Thus, resource-driven processes fulfill Rq_3.

In contrast, activity-oriented approaches for modeling and executing business processes (Section 2.3.1) provide only limited support for resources. The concept of resources is typically limited to resources capable of doing work, such as human actors and web services. The WS-HumanTask specification enables the addition of ad hoc attachments and comments, which can be considered a limited set of information resources [OAS10a]. Furthermore, Caramba enables the association of human resources and certain types of information resources with business processes [Dus04; Dus05]. However, these approaches are not designed to associate every category of resources with business processes such as a Wiki service or a web-based development environment. Thus, we claim that activity-oriented approaches meet requirement Rq_3 only partially. Similarly, constraint-based approaches (Section 2.3.2) do not provide extensive support for specifying resources participating in informal processes. The approach presented by Grondelle and Gülpers [vGG11] enables the definition of information resources and human resources in preconditions and postconditions of activities. However, this conceptual addition does not provide increased visibility of different types of resources other than information resources and human resources. Consequently, constraint-based approaches satisfy Rq_3 only partially.

Similarly, content-oriented approaches for modeling and executing business processes (Section 2.3.3) typically focus on information resources and human resources. In addition, the approach presented by Kumar and Wang [KW10] provides a means of modeling physical resources in business processes. However, the approach neglects IT resources capable of supporting actors [KW10]. Thus, we conclude that content-oriented approaches (Section 2.3.3) fulfill R_3 only partially.

3.4.4 Support for Dynamically Changing Resources (Rq_4)

Changing goals of informal processes results in changing resources (Section 3.1.4). Therefore, one of the characteristics of resource-driven processes is the support for dynamically changing resources. Human actors participating in business processes and business experts can update the set of resources in resource-driven process definitions during enactments of informal processes. Thus, resource-driven processes satisfy Rq_4.

For activity-oriented approaches for modeling and executing business processes (Section 2.3.1), adding new resources requires adding new activities into business process models. Similarly, activity-oriented approaches allowing ad hoc changes at runtime will enable dynamically changing resources by adding and removing activities, such as Caramba [Dus04; Dus05]. For example, to include a new Java developer in an informal process, actors need to add a new activity aimed at investigating the cause of a defect. Consequently, activity-oriented approaches for modeling and executing business processes partially satisfy Rq_4 due to their indirect support of dynamically changing resources.

Similarly, constraint-based approaches (Section 2.3.2) rely on activities for changing resources dynamically during enactments of informal processes. Thus, constraint-based approaches will fulfill Rq_4 only partially.

In contrast, content-oriented approaches for modeling and executing business processes (Section 2.3.3), such as case handling [vdAWG05] and adaptive case management [HK11; MS13; MSB+13], address Rq_4 by enabling changing information resources and human resources at runtime. Next, we present a discussion of the presented resource-driven process modeling and execution approach and Redo.

3.5 Discussion

Resource-driven processes enable support for actors of business processes and the reproducibility of their outcomes. Therefore, business experts document interrelated resources for achieving goals of their organizations in resource-driven process definitions (i.e., resource-driven process models). Furthermore, upon instantiating resource-driven process models, interrelated resources are automatically allocated. Thus, actors do not need to conduct activities for allocating resources. Moreover, modeling resources preserves the knowledge of previous enactments. For example, including a document with information regarding the execution of a business process will guide actors during future enactments. The resources included, such as a Git repository, will additionally facilitate better collaboration. In addition, to coordinate activities of human resources, activity-oriented business processes can be initialized in business processes. To regulate collaborations, business experts can use relationships. For example, relationships can be used to increase social collaborations by specifying teams with certain social structures proven in organizations, such as a flat hierarchy comprising former co-workers and a leader. Moreover, using relationships, organizations can restrict access to certain resources and provide implicit execution semantics to business processes. Interestingly, goals of business processes guide actors toward desired outcomes of business processes. Thus, describing the right resources will result in a resource set capable of achieving the goals of business processes, producing desired outcomes. Furthermore, resource-driven processes can be used to model every type of business process from the business process spectrum presented in Section 1.1 by relying on the "who" and "with" dimensions existing in every type of business process.

Redo presents a formal language capable of creating resource-driven process models based on their interrelated resources and goals. Resource-driven process definitions contain resource models with a graph of resources (vertices) and their relationships (edges). Therefore, the resource model resembles a topology template introduced in TOSCA Section 2.4.1 or an application instance model [Bin15], as explained in Section 2.4.1. Furthermore, node types and relationship types are equivalent for resource and relationship types. Node types represent application components, whereas resource types represent self-contained resources in an organization; therefore, they do not depend on the existence of other resources. Thus, resource types do not have requirements, as in the case of the optional requirements in TOSCA node types. In addition, goals present their requirements in terms of capabilities required by the goals. In contrast, both node types and resource types offer certain capabilities to their environments. Moreover, node types refer to implementation artifacts and deployment artifacts, enabling the execution of management operations on different application components.

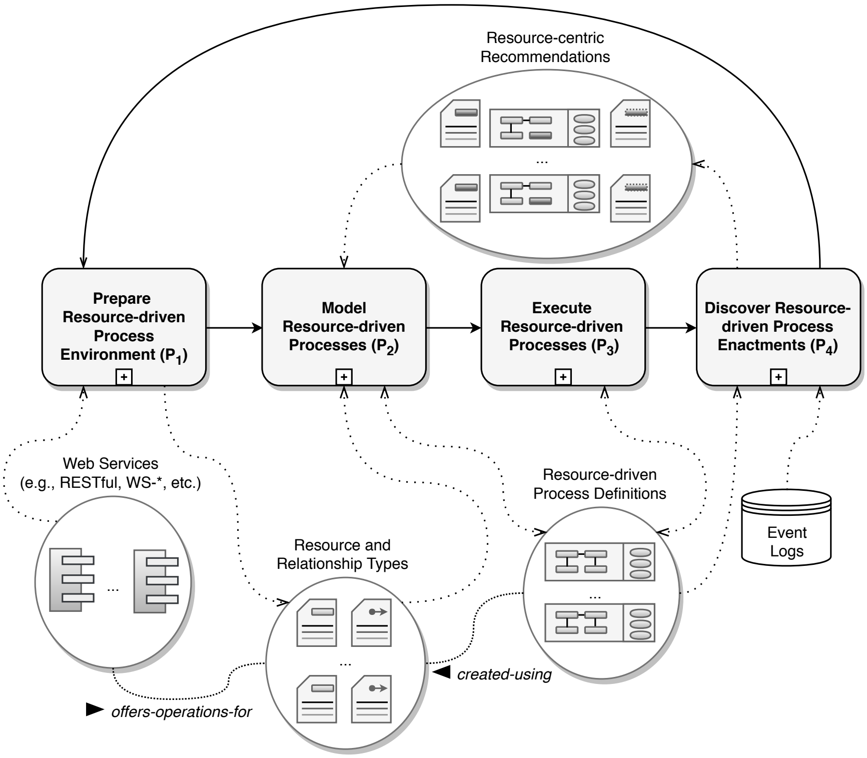
At a first glance, an operation endpoint can be conceptually similar to implementation artifacts and deployment artifacts of the TOSCA specification. However, there is a crucial difference; operation endpoints are web services and are always on, that is, they are not archived files ready for deployment, unlike implementation artifacts and deployment artifacts of the TOSCA specification. Furthermore, operation endpoints typically create an abstraction layer on top of existing resource management and automation systems. For example, an operation endpoint for allocating a MediaWiki resource can exploit the OpenTOSCA container or Docker behind the scenes. Thus, in its core concept, resource-driven processes foster the reuse of existing automated resource allocation approaches and their technologies.

Chapter 4

Using Resource-driven Processes in Organizations

The resource-driven process modeling and execution approach aims at achieving organizational goals by automatically allocating organizational resources. Although we described this approach in Section 3.2, its application in organizations is still an open research question that needs to be investigated. To answer this question, we present the Resource-driven Process Management (RPM) life cycle[1] enabling resource-driven process modeling and execution in organizations. The life cycle contains four phases aimed at (i) preparing IT infrastructure for resource-driven processes, (ii) modeling resource-driven processes (Contribution 2), (iii) executing resource-driven processes (Contribution 3), and (iv) generating recommendations for modeling based on the previous executions (Contribution 4). Consequently, the RPM life cycle presents the next three contributions of this work explained in Section 1.2.2. Next, we first present an overview of the RPM life cycle, summarizing the life

[1]We referred to the RPM life cycle as Informal Process Execution (InProXec) method in our former works.



Resource-centric Recommendations
...
...
Prepare Resource-driven Process Environment (P_1)
Model Resource-driven Processes (P_2)
Execute Resource-driven Processes (P_3)
Discover Resource-driven Process Enactments (P_4)
Web Services (e.g., RESTful, WS-*, etc.)
...
Resource-driven Process Definitions
...
Event Logs
Resource and Relationship Types
...
created-using
offers-operations-for

Figure 4.1: Overview of the Resource-driven Process Management life cycle.

cycle and its phases, so that the following sections are easier to comprehend. We present a conceptual framework detailing concepts involved in different phases of the RPM life cycle in Section 4.2. The conceptual framework is followed by a detailed explanation of the four phases of the RPM life cycle using the framework in Section 4.3.

4.1 Overview of the Resource-driven Process Management Life Cycle

The RPM life cycle comprises four phases with different steps to be executed in organizations by (i) technical and (ii) business experts. Therefore, we represent each phase as a sub-process construct of Business Process Model and Notation [Obj11], as shown in Figure 4.1. Next, we give an overview of each phase of the RPM life cycle, which will be detailed in the following.

Prepare Resource-driven Process Environment (P_1) By applying the RPM life cycle, organizations can model, execute, and improve their business processes in a resource-driven way. As each organization relies on their specific set of resources to achieve their organizational goals, during the first phase of the RPM life cycle, organizations adapt their modeling, execution, and discovery environments of resource-driven processes to their organizational needs. For example, a mobile device manufacturer will typically rely on both software experts and hardware experts in their business processes for fixing a defect in their product. In contrast, a software service provider will address a defect in their products by initially allocating their software experts. The main objectives of this phase are (i) making reusable types of resources and relationships of an organization available for modeling of resource-driven processes, (ii) automating the allocation of interrelated resources of resource-driven processes, and (iii) automating the discovery of process enactments. Thus, this phase is a preparation for the other phases and will involve software development activities, similar to the configuration phase of the Business Process Management (BPM) life cycle explained in Section 2.2. Business experts and technical experts analyze existing business processes and identify the set of resources and relationships required to adapt modeling, execution, and discovery of resource-driven processes to these resources. Furthermore, they can extend this set of resources and relationships based on their experience, intuition, and available resources and relationships of the respective organization. For example, a business expert and a technical expert of an organization that manufactures a mobile device will identify a set of resources and relationships including software developers, Git repositories, Java development environments, and Wiki services with "works-together," "updates," and "reads" relationships. The identification of resources and their relationships is followed by an analysis of the types of interactions, delivering insight on enactments of business processes. For example, "Git commit" interactions will provide insight on actual enactments of a business process by reflecting actual contributions of different actors.

After identifying resources, their relationships, and interactions providing insight about enactments of business processes, technical experts integrate software components capable of (i) making resources and their relationships available in resource-driven process modeling environments, (ii) automatically allocating interrelated modeled resources, and (iii) discovering enactments of resource-driven processes using interactions of resources in resource-driven process enactments. In this case, integrating means extending the modeling, execution, and discovery environments of resource-driven processes using the software components with the aforementioned three capabilities. For example, technical experts integrate their social networking service to make human resources available in their modeling environment of resource-driven processes. Thus, existing web services play an important role in this phase, as shown in Figure 4.1. Moreover, they create a web service interacting with human resources included in resource-driven process definitions to allocate them automatically upon initialization. Following the integration of these software components for modeling and execution environments of resource-driven processes, technical experts integrate software components, discovering enactments of resource-driven processes using interactions between resources, such as a software component analyzing Git interactions to discover software developers that participated in a resource-driven process. As a result of P_1, software components capable of (i) making resources and their relationships available in resource-driven process modeling environments and (ii) allocating interrelated resources automatically are created. Moreover, software components capable of discovering enactments of resource-driven processes can give hints to business experts when modeling resource-driven processes by identifying the resources used.

Model Resource-driven Processes (P_2) Organizations can use resource-driven processes to document both structured and unstructured business processes from the business process spectrum (Section 1.1). Therefore, during P_2, business experts create resource-driven process definitions, documenting interrelated resources and capabilities to accomplish goals of business processes. Thus, this phase can be considered the equivalent of the first phase of the BPM life cycle explained in Section 2.2. To create definitions of resource-driven processes, possible types of resources and their relationships should be available in the modeling environments of resource-driven processes. Consequently, business experts execute steps included in this phase after P_1, as shown in Figure 4.1. Business experts document interrelated resources and capabilities for achieving certain organizational goals. For example, business experts document that an informal process aimed at investigating and fixing a mobile device defect requires one software developer, two Git repositories, one Wiki service, and one project management service. Moreover, organizations achieve goals under certain context definitions, such as the presence of a mobile device defect, which is needed to achieve the goal of fixing the mobile device.

Accomplishing goals results in certain contexts, such as a functioning mobile device after achieving the goal of fixing it. To increase reusability and discoverability of goals, business experts document initial and final context definitions of goals. Furthermore, business experts associate goals with the capabilities that they require for their achievements, such as a software development capability needed to achieve the goal of fixing a mobile device defect. These goals are stored in resource-driven process definitions along with resource and relationship definitions for achieving goals. As a result of creating resource-driven process definitions, organizations can preserve their organizational knowledge regarding achieving their organizational goals based on their interrelated resources. Upon the initialization of resource-driven process definitions, interrelated resources are allocated automatically to achieve organizational goals, as explained next.

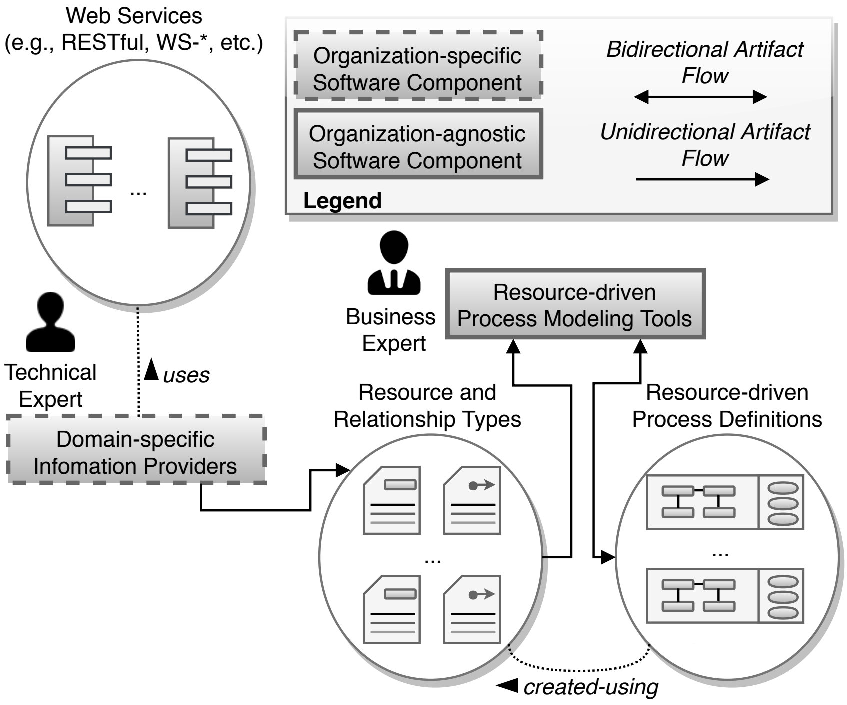
Execute Resource-driven Processes (P_3) Business experts create resource-driven process definitions not only to document business processes in a resource-driven way but also to provide a degree of automation and support for the business processes. Therefore, during P_3, business experts start resource-driven processes, resulting in an automated allocation of interrelated documented resources. This phase is directly executed after P_2, as the phase starts with initializing resource-driven process definitions. Thus, this phase can be considered equivalent to the third phase of the BPM life cycle explained in Section 2.2. Upon the initialization of resource-driven process definitions, execution environments of resource-driven processes allocate resources and establish their relationships using the software components integrated during P_1. For example, during the allocation of a human resource of a resource-driven process, a software service integrated during P_1 invites the actor to participate in the resource-driven process. If the actor agrees to participate, a relationship between the actor and a Wiki service is established by executing a business logic adding a new user to the Wiki service. After initializing a resource-driven process, allocated resources collaboratively and autonomously work toward goals of resource-driven processes. Furthermore, completing enactments of resource-driven processes successfully or unsuccessfully results in an automated deallocation of the allocated resources. As a result of P_3, interaction traces are left by resources that have worked collaboratively to achieve goals specified in resource-driven process definitions. These interaction traces of resources enable the possibility of discovering actual enactments, as explained next.

Discover Resource-driven Process Enactments (P_4) Goals or the priority of goals of business processes typically change during their enactments in P_3. To adapt to these changes, actors of resource-driven processes typically engage in ad hoc collaborations. As a result, resources that are not yet documented in resource-driven process definitions can make contributions to enactments of resource-driven processes. For example, an external software developer who is consulted in an ad hoc fashion during an enactment of a resource-driven process can change the source code of the software used in a mobile device produced by the organization. Consequently, during future enactments of the same resource-driven process, this software developer becomes relevant due to historical contributions. Obviously, including contributing resources or their required capabilities for the contributions in definitions of resource-driven processes can improve the performance of their enactment, such as shorter completion time and better outcomes. For example, if a defect is caused by a change made by an external expert, allocating this resource from the beginning of a resource-driven process will improve the efficiency of this process. Moreover, not all resources documented in resource-driven process definitions make contributions to resource-driven processes. For example, it is possible that a software developer of a certain programming language documented in a resource-driven process definition could make no contribution. Thus, including this resource in future executions can result in a waste of organizational resources.

Providing information regarding the degree of contributions of resources will support the decision-making process of business experts when selecting different resources for resource-driven processes. Therefore, software components that were integrated during P_1 analyze interactions of allocated resources specified in resource-driven process definitions in event logs to discover actual executions of resource-driven processes after initializing the resource-driven process definitions. For example, to identify software developers and the amount of the contribution made by them, the software components analyze the Git interactions made in a Git repository of the respective resource-driven process. To this end, the type of interactions collected depends on the type of the resource being analyzed, such as Git interactions and email interactions for the resource type "software developer." Consequently, this phase can be considered equivalent to the fourth phase of the BPM life cycle explained in Section 2.2. After this analysis, resources and capabilities are discovered in the form of recommendations (i.e., resource-centric recommendations), providing insight regarding enactments of resource-driven processes. We consider these only recommendations because they are facts relying on the limited information available in a limited set of interactions identified at P_1. For example, recommendations will not include the information available in an email, if business experts and technical experts decide not to do so in P_1. Thus, the generated recommendations can be biased by focusing on only a specific type of interaction. As a result of this phase, generated recommendations can give business experts hints to ease their modeling process based on the information from actual enactments of resource-driven processes. Moreover, including contributing resources can provide better performance metrics, such as shorter completion times or better process outcomes. Next, we detail concepts to execute the RPM life cycle.

4.2 Conceptual Framework of the Resource-driven Process Management Life Cycle

The conceptual framework presented in this section consists of different software components involved in different phases of the RPM life cycle. We distinguish between organization-specific and organization-agnostic software components. Organization-specific software components refer to the software components that need to be adapted based on the needs of organizations, such as adding an organization-specific software component capable of allocating physical resources in a production company. In contrast, organization-agnostic software components can be used across different components without adaptation steps. Moreover, phase P_1 aims at adapting different organization-specific software components from different phases to organizational needs. Next, we explain software components based on the phase in which they are used, that is P_2, P_3, and P_4. Consequently, we do not present a separate section for the preparation phase (P_1), because none of the components is used in this phase. After explaining the conceptual framework, we detail the phases using the conceptual framework in Section 4.3.



Web Services (e.g., RESTful, WS-*, etc.)
Organization-specific Software Component
Organization-agnostic Software Component
Legend
Bidirectional Artifact Flow
Unidirectional Artifact Flow
Business Expert
Resource-driven Process Modeling Tools
Technical Expert
uses
Domain-specific Infomation Providers
Resource and Relationship Types
Resource-driven Process Definitions
created-using

Figure 4.2: Illustration of a resource-driven process modeling environment.

4.2.1 Conceptual Framework of Modeling Resource-driven Processes

Creating resource-driven process definitions relying on interrelated resources requires the availability of different types of resources and their possible relationships in modeling environments of resource-driven processes. For example, business experts specify a resource-driven process definition aimed at fixing a mobile product defect using interrelated resources, such as a software developer, a Git repository, a project management tool, and a Wiki service. Figure 4.2 presents an illustration of such a resource-driven process modeling environment. Moreover, different types of organizations can create resource-driven process definitions with different types of resources. For example, an organization manufacturing a product using a 3-D printer will need to represent this printer in their definitions of resource-driven processes, so that the 3-D printer can be allocated on-demand. In this case, allocating the 3-D printer means reserving the machine for a resource-driven process. In many other organizations, such a resource will have no use. Consequently, organizations need to make their custom set of resources available in their modeling environments of resource-driven processes. For this purpose, during P_1, technical experts develop domain-specific information providers. In the following, we make use of the definitions introduced in Resource-driven Process Modeling Language (Redo).

Definition 30 (Domain-specific Information Provider). Let a domain-specific information provider be $dip \in DIP$. Each domain-specific information provider is a triple defined as follows:

$$DIP \subseteq IE \times \wp(\mathcal{S}) \times \wp(\mathcal{R})$$

$$dip = (ie, \mathcal{S}_{dip}, \mathcal{R}_{dip})$$

Where:

- $\pi_1(dip) = ie$, $ie \in IE$ is an identifiable entity definition specifying the represented domain-specific information provider,
- $\pi_2(dip) = \mathcal{S}_{dip}$, $\mathcal{S}_{dip} \subseteq \wp(\mathcal{S})$ are resource types provided by this domain-specific information provider,
- $\pi_3(dip) = \mathcal{R}_{dip}$, $\mathcal{R}_{dip} \subseteq \wp(\mathcal{R})$ are relationship types provided by the domain-specific information provider. ♣

Conceptually, domain-specific information providers deliver standard representations of the resources and relationships available in organizations: resource types (Definition 21) and relationship types (Definition 23), as shown in Figure 4.2. For example, a domain-specific information provider of IT resources can use an OpenTOSCA container (Section 2.4.1) to list available services deployed in the container and present these as resource types. Thus, domain-specific information providers use other web services, if applicable. Depending on organizational needs, organizations can select a custom set of domain-specific information providers during P_1, as detailed in Section 4.3.1. Consequently, domain-specific information providers are considered organization-specific software components. Using these resource and relationship types, business experts can create resource-driven process definitions (Definition 29) containing resources and relationships in their organizations. To create resource-driven process definitions, business experts use resource-driven process modeling tools.

Definition 31 (Resource-driven Process Modeling Tool (informal)). Resource-driven process modeling tools are modeling tools capable of creating models defined in Resource-driven Process Modeling Language. ♣

Conceptually, resource-driven process modeling tools enable a means of documenting organizational practices to achieve organizational goals in a resource-driven way. More specifically, business experts use resource-driven process modeling tools to create resource-driven process definitions using resource and relationship types provided by domain-specific information providers, as shown in Figure 4.2. Furthermore, resource-driven process modeling tools are used to edit modeling elements specified in an organizational definition (Definition 9), such as context definitions, goal definitions, capability definitions, and resource-driven process definitions defined in an organizational definition. Details of modeling steps using these tools will be presented in Section 4.3.2. Moreover, resource-driven process modeling tools are typically organization-agnostic software components and are not customized depending on the nature of an organization.

In Redo, many modeling elements refer to each other using entity identities. Thus, it is typically necessary to determine the represented modeling element by an entity identity, such as a resource-driven process definition or a resource definition represented by an entity identity. Therefore, the entity identity resolver function is used.

Definition 32 (Entity Identity Resolver). Let an entity identity resolver be a following function:

$$\sigma_{ME} : ID_{od} \rightarrow ME_{od} \cup \{\epsilon\}$$

Where:

- $dom(\sigma_{ME}) = ID_{od}$ is the set of entity identities defined in an organizational definition od containing $id \in ID_{od}$,
- $img(\sigma_{ME}) = ME_{od} \cup \{\epsilon\}$ is the set of modeling elements defined in an organizational definition od and ϵ representing unavailability of a modeling element with the given entity identity in the organizational definition, such that every modeling element $me \in img(\sigma_{ME})$ is uniquely identified by an entity identity $id \in dom(\sigma_{ME})$. ♣

The function maps entity identities to modeling elements in an organizational definition (Definition 9). For example, the function will return a context definition representing the presence of a defect of a mobile device by providing the respective entity identity of the definition. In case no such modeling element is in the corresponding organizational definition, the function returns ϵ. As each entity identity maps to a modeling element or to ϵ, the function is surjective. A combination of an organizational definition, domain-specific information providers, resource-driven process modeling tools, and an entity identity resolver comprise the resource-driven process modeling environment. Formally, resource-driven process modeling environments are defined as follows:

Definition 33 (Resource-driven Process Modeling Environment). Let a resource-driven process modeling environment be $ms \in MS$. Each resource-driven process modeling environment is a quadruple defined as follows:

$$ms = (od, \mathcal{P}M_{ms}, DIP_{ms}, \sigma_{ME})$$

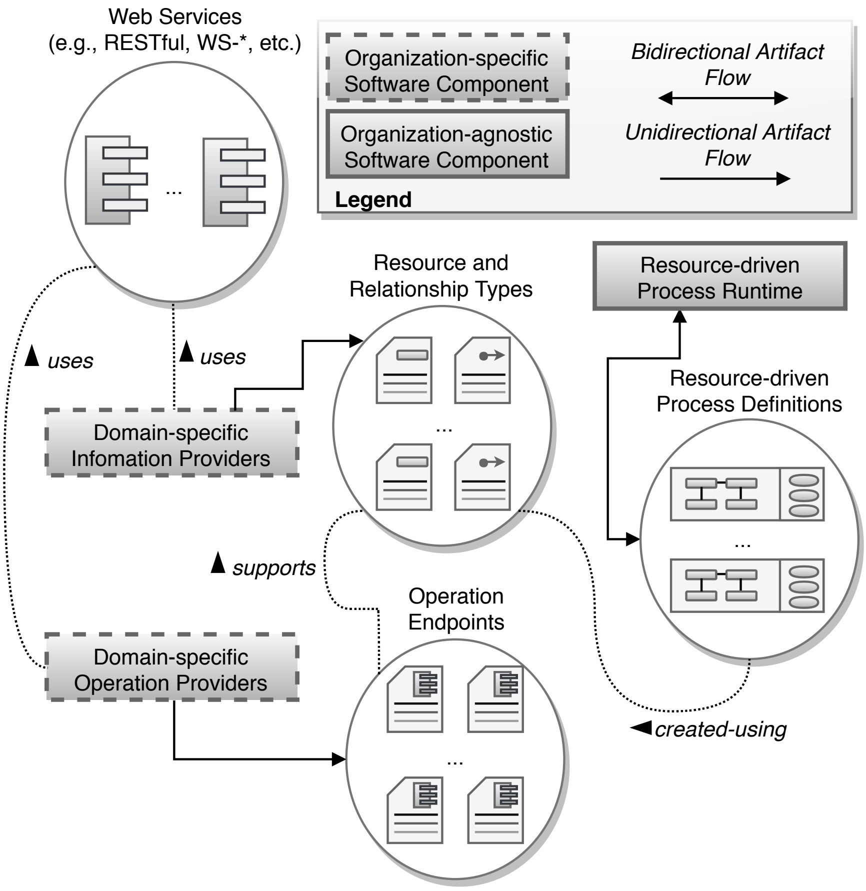
Where:

- $\pi_1(ms) = od$, $od \in OD$ is an organizational definition containing modeling elements available in the resource-driven process modeling environment (Definition 9),
- $\pi_2(ms) = \mathcal{PM}_{ms}$, $\mathcal{PM}_{ms} \subseteq \wp(\mathcal{PM})$ is a set of resource-driven process modeling tools, such that $1 \leq |\mathcal{PM}_{ms}|$ (Definition 31),
- $\pi_3(ms) = DIP_{ms}$, $DIP_{ms} \subseteq \wp(DIP)$ is the set of domain-specific information providers, such that $1 \leq |DIP_{ms}|$ (Definition 30),
- $\pi_4(ms) = \sigma_{ME}$ is a function mapping from an entity identity of a modeling element to the modeling element (Definition 32). ♣

Conceptually, a resource-driven process modeling environment represents concepts enabling the creation of resource-driven process definitions (Definition 29), as detailed in Section 4.3.2. As the main objective of resource-driven process modeling environments is creating resource-driven process definitions (Definition 29), it requires at least one resource-driven process modeling tool and one domain-specific information provider to deliver a set of resource and relationship types, as shown in Figure 4.2. In the absence of domain-specific information providers, there will not be any reusable resource and relationship types to create resource-driven process definitions. After creating resource-driven process definitions, business experts initialize these to achieve goals specified in resource-driven process definitions, resulting in the execution of resource-driven processes. Next, we detail the conceptual framework of executing resource-driven processes.

4.2.2 Conceptual Framework of Executing Resource-driven Processes

Resource-driven processes involve the automated allocation of interrelated resources upon initializing resource-driven process definitions. Therefore, organizations employ resource-driven process execution environments, as illustrated in Figure 4.3. For example, if business experts create a resource-driven process definition containing a Wiki service in P_2, the Wiki service can be automatically allocated in the IT infrastructure of the respective



Web Services (e.g., RESTful, WS-*, etc.)
Organization-specific Software Component
Organization-agnostic Software Component
Bidirectional Artifact Flow
Unidirectional Artifact Flow
Legend
Resource and Relationship Types
Resource-driven Process Runtime
uses
uses
Domain-specific Infomation Providers
Resource-driven Process Definitions
supports
Operation Endpoints
Domain-specific Operation Providers
created-using

Figure 4.3: Illustration of a resource-driven process execution environment.

organization. To enable the automated allocation of interrelated resources specified in resource-driven process definitions, technical experts develop domain-specific operation providers capable of providing operation endpoints (Definition 20), which are representations of operations in Redo.

The operations represented by operation endpoints typically support only a limited set of resource and relationship types. Supporting, in this context, means the operation provided by an operation endpoint can be executed on a limited set of resource or relationship types. For example, an operation allocating IT services using the OpenTOSCA container (Section 2.4.1) will support the set of resource types representing IT resources. In contrast, such an operation will not support resource types representing human resources. To bind resource and relationship types with operations supporting them, operation bindings are used:

Definition 34 (Operation Binding). Let an operation binding be $oeb \in OEB$. Each operation binding is a pair defined as follows:

$$OEB \subseteq OE \times \wp(ID_{od})$$

$$oeb = (oe, ID_{sr})$$

Where:

- $\pi_1(oeb) = oe$, $oe \in OE$ is the operation endpoint specified,
- $\pi_2(oeb) = ID_{sr}$, $ID_{sr} \subseteq \wp(ID_{od})$ are entity identities referring to resource types or relationship types that the operation endpoint oe supports. ♣

Conceptually, each operation binding describes resource and relationship types supported by an operation endpoint specified in the operation binding. Operation endpoints enable the execution of different operations on organizational resources and relationships, such as executing life cycle operations during P_3. In the context of this work, we distinguish between four types of life cycle operations for different entities, such as resources and relationships. Formally, a subset of entity identities is used to specify different types of life cycle operations:

Definition 35 (Life Cycle Operation Type). Let a life cycle operation type be $id_{lifecycle} \in ID_{lifecycle}$. Life cycle operation types are a subset of entity identities $ID_{lifecycle} \subseteq ID$, such that the *namespace* of each life cycle operation type is $LIFECYCLE\_NAMESPACE$ and the *name* of each life cycle operation type is in {ALLOCATE, DEALLOCATE, ESTABLISH, RELEASE}. ♣

Conceptually, each life cycle operation type is specified under a predefined namespace and possible different operation names exist depending on the modeling element. For example, resources are “allocated” and “deallocated” and resource relationships are “established” and “released”. Using life cycle operation types, a life cycle operation endpoint can be specified formally:

Definition 36 (Life Cycle Operation Endpoint). Let a life cycle operation endpoint be $oe_{lifecycle} \in OE_{lifecycle}$. Life cycle operation endpoints are subsets of operation endpoints $OE_{lifecycle} \subseteq OE$, such that the operation type of a life cycle operation endpoint is a life cycle operation type. ♣

Each life cycle operation endpoint is an operation endpoint with restricted operation types, that is, its operation type is in the set of life cycle operation identities. In addition to the life cycle operations, operation endpoints can be any operation that can be executed on certain types of resources and relationships. For example, an operation endpoint can provide an “add a new administrator” operation to enable the automated establishment of a “managing” relationship between a human resource and a Wiki service. Similarly, operation endpoints can provide different artifacts, providing information regarding a resource type. For example, an operation endpoint can provide CSAR files of TOSCA resources using a Winery backend. Domain-specific operation providers are composed of one or more operation bindings and each domain-specific operation provider is defined as follows:

Definition 37 (Domain-specific Operation Provider). Let a domain-specific operation provider be $dop \in DOP$. Each domain-specific operation provider is a pair defined as follows:

$$DOP \subseteq IE \times \wp(OEB)$$

$$dop = (ie, OEB_{dop})$$

Where:

- $\pi_1(dop) = ie$, $ie \in IE$ is an identifiable entity definition specifying the represented domain-specific operation provider,

- $\pi_2(dop) = OEB_{dop}$, $OEB_{dop} \subseteq \wp(OEB)$ are operation bindings delivered by the respective domain-specific operation provider *dop*. ♣

Conceptually, domain-specific operation providers group a set of operation endpoints available in a domain of resources, as shown in Figure 4.3. For example, a domain-specific operation provider of TOSCA resources can group all different operation bindings supporting resource and relationship types that can be deployed on the OpenTOSCA container (Section 2.4.1). Similarly, another domain-specific operation provider can provide all available operation bindings of resource types and relationship types addressing resources deployed on Docker (Section 2.4).

Organizations will typically rely on multiple domain-specific information and domain-specific operation providers (i) to represent resources and relationships from different domains and (ii) to automate allocation and deallocation of resources and relationships from different domains. For example, organizations can create domain-specific information and domain-specific operation providers for the domain of human resources and IT resources separately. Resource-driven process runtimes exploit operation endpoints provided by domain-specific operation providers to manage life cycles of interrelated resources during resource-driven process executions:

Definition 38 (Resource-driven Process Runtime (informal)). Resource-driven process runtimes are software components capable of managing instances of resource-driven processes modeled in Resource-driven Process Modeling Language. ♣

Resource-driven process runtimes are software components capable of initializing and finalizing resource-driven process executions by allocating and deallocating interrelated resources specified in resource-driven process definitions automatically, as detailed in Section 4.3.3. To store the state information of executions of a resource-driven process, they make use of instance descriptors of definitions for resource-driven processes, resources, and relationships. To allocate interrelated resources automatically, resource-driven process runtimes must select the right operation endpoint specified under resource and relationship types. For example, to allocate a Wiki service, an operation endpoint supporting the allocation of the Wiki service should be selected out of all available endpoints. Therefore, the following selection function is used:

Definition 39 (Operation Endpoint Selector). Let an operation endpoint selector be a function:

$$\sigma_{OE} : ID_{od} \times ID \rightarrow OE \cup \{\epsilon\}$$

Where:

- $dom_1(\sigma_{OE}) = ID_{od}$ is the set of entity identities referring to a resource or relationship type in an organizational definition *od*,
- $dom_2(\sigma_{OE}) = ID$ is the set of entity identities referring to the operation type of an operation endpoint,
- $img(\sigma_{OE}) = OE \cup \{\epsilon\}$ is the set of operation endpoints *OE* and ϵ, such that for each operation endpoint $oe_{img} \in img(\sigma_{OE})$ and the operation type of operation endpoint $\pi_2(oe_{img}) = id_{operationType}$:
  - The operation endpoint oe_{img} is in the set of operation endpoints of the resource or relationship type referred to by the entity identity $id \in dom_1(\sigma_{OE})$,
  - The operation type $id_{operationType}$ is in the second domain of the operation endpoint selector function $id_{operationType} \in dom_2(\sigma_{OE})$,
  - Otherwise, the function maps to ϵ, meaning no such operation endpoint is available. ♣

The function takes two input parameters specifying (i) an entity identity of resource or relationship types in an organizational definition and (ii) an entity identity representing the type of desired operation endpoint. Moreover, the function returns an operation endpoint that (i) contains an operation type ($id_{operationType}$) specified by the second input parameter and (ii) is part of the resource or relationship types referred to by the first input parameter. For example, using the function, a resource-driven process runtime can select an operation endpoint of an allocation operation of a resource type representing a Wiki service. Therefore, the entity identity of the resource type describing the Wiki service and the entity identity specifying an allocation operation should be given to the function.

While initializing resource-driven process definitions, not only are resources automatically allocated but relationships are also automatically established. For example, allocating a human resource requires establishing its relationships, such as a managing relationship regarding a Wiki service. To retrieve relationship definitions of a resource definition, the following function is used:

Definition 40 (Relationship Retriever). Let a relationship retriever be a function:

$$\sigma_{RD} : SD_{od} \times RM_{od} \rightarrow \wp(RD_{od})$$

Where:

- $dom_1(\sigma_{RD}) = SD_{od}$ is the set of resource definitions in an organizational definition *od*, such that each resource definition *sd* in the second domain of the relationship retriever ($sd \in dom_1(\sigma_{RD})$) is a vertex in a resource model ($rm \in dom_2(\sigma_{RD})$),
- $dom_2(\sigma_{RD}) = RM_{od}$ is the set of resource models in an organizational definition *od*,
- $img(\sigma_{RD}) = \wp(RD_{od})$ is the power set of relationship definitions in an organizational definition *od*, such that the target or source resource definition referred to by every relationship definition *rd* in the range of the relationship retriever ($rd \in RD_{od} \in img(\sigma_{RD})$) is in the first domain of the relationship retriever function ($dom_1(\sigma_{RD})$). ♣

The function takes a resource definition and the resource model containing the resource definition is given as the second parameter. The function maps these parameters to relationship definitions of the resource definition in the resource model given as the first parameter. As described in Definition 40, every output relationship definition refers to the resource definition of the input resource definition as the source or target resource definition. Checking relationships can be necessary during the allocation of resources, as additional business logic can be needed to establish a certain relationship during or after allocating a resource, such as a business logic (i) checking the satisfaction of a "has-worked-together" relationship before allocating a human resource and (ii) adding an admin to a Wiki service for establishing a "managing" relationship after allocating a human resource. Similarly, before and after deallocating resources, it can be required to execute a cleaning operation associated with relationship definitions. For example, during the deallocation of a Wiki service, organizations typically will not want to erase their collected knowledge. To avoid this, they will suspend the Wiki service without deleting the data it contains. During the next execution, the Wiki service will be reallocated with the preserved data. Moreover, during the next execution, different human resources can participate in comparison with the previous one. Therefore, all added users should be removed before deallocating a Wiki service. Operation endpoints contained in relationship types are used for executing such operations. Selecting the right operation endpoint of a relationship type can be done using the operation endpoint selector function, as detailed in Section 4.3.3.

Resource and relationship definitions can be initialized in the context of different instances of the same resource-driven process definition. Hence, resource and relationship definitions will contain multiple instance descriptors referring to different instances of resource-driven processes. To retrieve the instance descriptors created in the context of a parent instance, such as an instance descriptor of a resource definition in a resource-driven process definition, we define the following function:

Definition 41 (Child Instance Descriptor Retriever). Let a child instance descriptor retriever be a function:

$$\sigma_{\mathcal{I}} : ID_{od} \times ID_{od} \rightarrow \mathcal{I}_{od} \cup \{\epsilon\}$$

Where:

- $dom_1(\sigma_{\mathcal{I}}) = ID_{od}$ is the set of entity identities referring to a set of definitions for resources, relationships, goals, or resource-driven processes in an organizational definition *od*,
- $dom_2(\sigma_{\mathcal{I}}) = ID_{od}$ is the set of entity identities referring to instance descriptors in an organizational definition *od*,
- $img(\sigma_{\mathcal{I}}) = \mathcal{I}_{od} \cup \{\epsilon\}$ is a set of instance descriptors and ϵ, such that:
  - The entity identity referring to the source model $id_{sourceModel}$ of every instance descriptor $i \in img(\sigma_{\mathcal{I}})$ is in the domain set of the first input parameter $id_{sourceModel} \in dom_1(\sigma_{\mathcal{I}})$,
  - The entity identity referring to the parent model id_{parent} of every instance descriptor $i \in img(\sigma_{\mathcal{I}})$ is in the domain set of the second input parameter $id_{parent} \in dom_2(\sigma_{\mathcal{I}})$,
  - Otherwise the function maps to ϵ, meaning no such instance descriptor is available. ♣

The child instance descriptor retriever function returns an instance descriptor of a resource, relationship, or goal definition based on the id_{parent} of instance descriptors. More specifically, the first input of the function refers to the modeling element containing the resulting instance descriptor, and the second input parameter is an entity identity specifying the id_{parent} of the resulting instance descriptor. Consequently, the returned instance descriptor will contain the first input parameter of the function in its $id_{sourceModel}$ field and the second input parameter in its id_{parent} field. For example, the function will map an entity identity referring to a resource definition and an entity identity referring to a resource-driven process definition to the instance descriptor of the resource definition created in the context of the respective resource-driven process. As a result, the child instance descriptor retriever function can be used to retrieve a specific instance descriptor of a resource, relationship, or goal definition created in the context of a specific resource-driven process.

Including an organizational definition, a resource-driven process runtime, and domain-specific operation providers, the aforementioned functions in an organization enable automatically initiating executions of resource-driven processes by automatically allocating desired interrelated resources for specified goals. Moreover, as shown in Figure 4.3, this combination makes up a resource-driven process execution environment:

Definition 42 (Resource-driven Process Execution Environment). Let a resource-driven process execution environment be $es \in ES$. Each resource-driven process execution environment is an eight-tuple defined as follows:

$$es = (od, pc, DIP_{es}, DOP_{es}, \sigma_{OE}, \sigma_{RD}, \sigma_{\mathcal{I}}, \sigma_{ME})$$

Where:

- $\pi_1(es) = od$, $od \in OD$ is an organizational definition containing modeling elements available in the resource-driven process execution environment (Definition 9),
- $\pi_2(es) = pc$, $pc \in \mathcal{PC}$ is a resource-driven process runtime (Definition 38),
- $\pi_3(es) = DIP_{es}$, $DIP_{es} \subseteq \wp(DIP)$ is the set of domain-specific information providers available in an organization, such that $1 \leq |DIP|$ (Definition 30),
- $\pi_4(es) = DOP_{es}$, $DOP_{es} \subseteq \wp(DOP)$ is the set of domain-specific operation providers, such that $1 \leq |DOP|$ (Definition 37),
- $\pi_5(es) = \sigma_{OE}$ is a function mapping from a resource type and an operation type referred to by two entity identities to an operation endpoint supporting the resource type in the given operation type (Definition 39),
- $\pi_6(es) = \sigma_{RD}$ is a function mapping from a resource definition and a resource model to a set of relationship definitions indicating the resource definition as their source and target resource definition (Definition 40),

- $\pi_7(es) = \sigma_{\mathcal{I}}$ is a function mapping from an entity identity referring to a resource, relationship, or goal definition and an entity identity indicating a parent instance descriptor to a child instance descriptor representing an instance of the modeling element referred to by the first parameter and initialized in the context of an instance described by the second parameter (Definition 41),
- $\pi_8(es) = \sigma_{ME}$ is a function mapping from an entity identity of a modeling element to the modeling element (Definition 32). ♣

Conceptually, resource-driven process execution environments are responsible for automatically initializing and finalizing resource-driven process executions, as detailed in Section 4.3.3. Therefore, in each resource-driven process execution environment, there exist at least (i) one domain-specific information provider and (ii) one domain-specific operation provider delivering (i) available resource and relationship types and (ii) operation endpoints supporting these resources and relationships, as shown in Figure 4.3. The organizational definition residing in a resource-driven process execution environment preserves available resource types, relationship types, and operation endpoints supporting these. During initializing and finalizing a resource-driven process, the resource-driven process runtime automatically makes use of the information stored in the organizational definition to allocate and deallocate interrelated resources specified in a resource-driven process definition, as detailed in Section 4.3.3. Furthermore, the functions are used during allocation and deallocation of interrelated resources. Typically, different organizations do not customize resource-driven process runtimes. Thus, these are organization-agnostic software components. In contrast, different organizations typically use different sets of domain-specific operation providers, as shown in Figure 4.3 and described in Section 4.3.1. Each instance of resource definitions refers to real-world resources involved in the corresponding resource-driven processes (e.g., a developer and a Wiki service). Moreover, instance descriptors of resource definitions provide the required information for identifying and analyzing interactions during the execution of resource-driven processes. Interestingly,

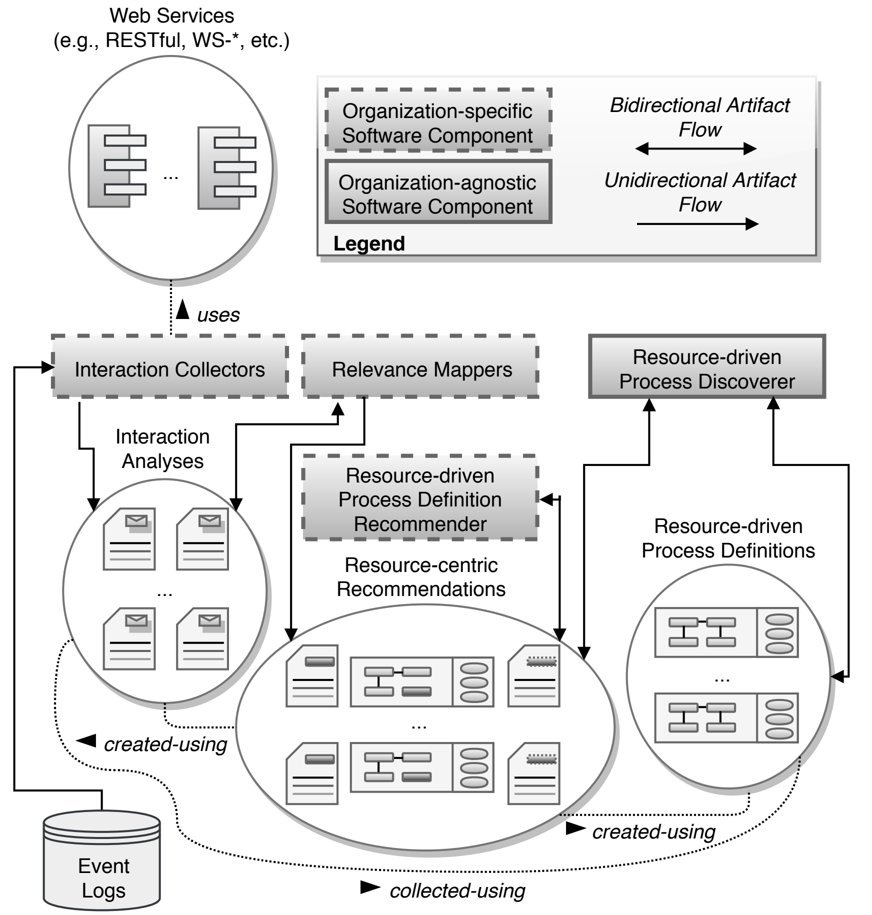

Web Services
(e.g., RESTful, WS-*, etc.)
Organization-specific Software Component
Organization-agnostic Software Component
Bidirectional Artifact Flow
Unidirectional Artifact Flow
Legend
uses
Interaction Collectors
Relevance Mappers
Resource-driven Process Discoverer
Interaction Analyses
Resource-driven Process Definition Recommender
Resource-centric Recommendations
Resource-driven Process Definitions
created-using
created-using
collected-using
Event Logs

Figure 4.4: Illustration of a resource-driven process discovery environment.

such an analysis can present insight about actual executions of resource-driven processes. Next, we provide the conceptual framework for generating recommendations based on interactions between resources.

4.2.3 Conceptual Framework of Discovering Resource-driven Process

After the initialization of resource-driven process definitions, allocated resources work toward goals of resource-driven processes. As business processes can involve dynamically changing goals (Section 2.1.1), allocated resources of resource-driven processes can engage in ad hoc collaborations to address newly emerging goals. These ad hoc collaborations can involve additional resources that were not included in resource-driven process definitions of the respective processes. Moreover, resources participating in these ad hoc collaborations or their capabilities can play an important role, when business processes aimed at the same goals are executed again. For example, the allocated resources of a resource-driven process for fixing a mobile device defect can collaborate with an expert of the device operating system. If this expert changes mobile device software, he or she can provide resourceful information in the future executions of resource-driven processes for fixing the mobile device defect. Thus, including this expert in the process from the beginning can reduce the time to resolve the mobile device defect. Similarly, a certain capability of such an expert can be useful for future executions aimed at investigating and fixing a mobile device defect, such as an Android development capability used by the expert. To understand the involvement of such resources and capabilities, interactions made by participating resources can be analyzed. Analyzing interactions requires collecting interactions of resources, which are collected by interaction collectors.

Definition 43 (Interaction Collector (informal)). Interaction collectors are software components capable of delivering interactions between a given resource and other resources from event logs using the instance descriptor of the given resource. ♣

Each interaction collector is a software component using available event logs in organizations to collect interactions of resources occurring during executions of business processes, as shown in Figure 4.4. An example of an event log is a Git repository storing interactions of development activities. Thus, gathering interactions requires the use of Application Programming

Interfaces (APIs) of other IT resources, such as the REST interface of a Git service. Using these interactions, it is possible to inspect different aspects of business processes, such as conducted activities, their sequence, and resources conducting these activities. As the resource-driven processes do not explicitly focus on modeling and executing interrelated activities but focuses on resources and their capabilities, we will investigate these based on collected interactions. To conduct such an analysis, interaction collectors structure each collected interaction as follows:

Definition 44 (Interaction Analysis). Let an interaction analysis be $ia \in IA$. Each interaction analysis is a quintuple:

$$IA \subseteq IE \times \mathcal{I}_{od} \times \mathcal{I}_{od} \times \Sigma^{+} \times \mathbb{Z}^{+}$$

$$ia = (ie, i_{analyzedResource}, i_{relevantResource}, interactionContent, iterationCount)$$

Where:

- $\pi_1(ia) = ie$, $ie \in IE$ is an identifiable entity definition specifying the represented interaction,
- $\pi_2(ia) = i_{analyzedResource}, i_{analyzedResource} \in \mathcal{I}_{od}$ is an instance descriptor in an organizational definition *od* representing an analyzed resource (i.e., the instance descriptor of a resource definition used to create the interaction analysis),
- $\pi_3(ia) = i_{relevantResource}, i_{relevantResource} \in \mathcal{I}_{od}$ is an instance descriptor in an organizational definition *od* representing an instance of a relevant resource,
- $\pi_4(ia) = interactionContent, interactionContent \in \Sigma^{+}$ is the content of the represented interaction (e.g., a Git commit or an email interaction),
- $\pi_5(ia) = iterationCount, iterationCount \in \mathbb{Z}^{+}$ is the iteration count of the represented interaction. ♣

Conceptually, an interaction analysis is a result of analyzing a resource for its interactions with other resources in event logs, as shown in Figure 4.4. This

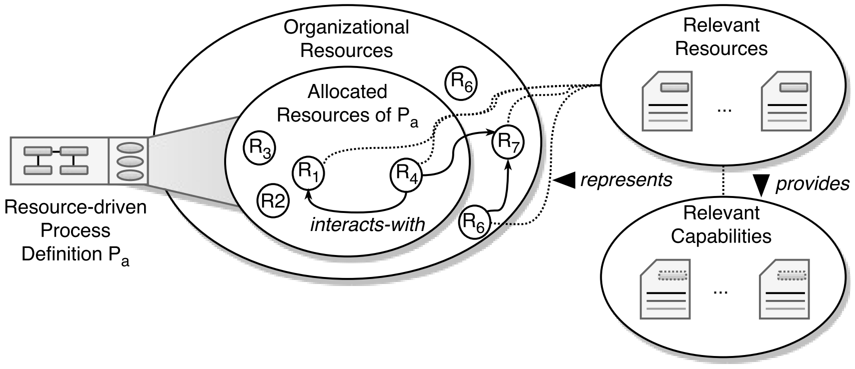


Figure 4.5: Illustration of relevant resources and relevant capabilities.

analysis starts using the resources allocated for a certain resource-driven process. Consequently, interaction collectors use instance descriptors (Definition 16) in resource models (Definition 26) of the corresponding resource-driven process definition to gather interactions of allocated resources. In this work, we call each identified resource type during the analysis of a resource for its interactions a relevant resource:

Definition 45 (Relevant Resource (informal)). A relevant resource is a resource type identified in an interaction analysis collected by an interaction collector during the analysis of another resource. ♣

Figure 4.5 presents an illustration of relevant resources and relevant capabilities. Moreover, each resource interacting with an allocated resource is considered an instance of a relevant resource, such as R_1, R_4, and R_7. For example, if an interaction collector identifies a developer after investigating interactions of a Git repository, the resource type representing the developer is identified as a relevant resource. Furthermore, resources interacting with identified instances of relevant resources are considered instances of relevant resources too, such as R_6 interacting with R_7 in Figure 4.5. To derive resource types, instance descriptors of identified resources are used. Therefore, in an interaction analysis, we represent an analyzed resource and instance of a relevant resource using instance descriptors. Moreover, each interaction analysis associates an analyzed resource with an instance of a relevant resource, even if the actual interaction contains multiple instances of relevant resources. If an interaction, such as an email with multiple recipients, contains multiple instances of relevant resources, for each instance of a relevant resource, an interaction analysis is created. The reason behind containing each instance of a relevant resource separately is having a one-to-one mapping between each instance and its source interaction. To generate the interaction analysis of an allocated resource represented by an instance descriptor using an interaction collector, the following function is used:

Definition 46 (Interaction Analysis Generator). Let an interaction analysis generator be a function:

$$gen_{IA} : IC \times \mathcal{I} \rightarrow \wp(IA)$$

Where:

- $dom_1(gen_{IA}) = IC$ is a set of interaction collectors,
- $dom_2(gen_{IA}) = \mathcal{I}$ is the set of instance descriptors representing instances of a set of resource definitions,
- $img(gen_{IA}) = \wp(IA)$ is the power set of interaction analyses, such that the analyzed resources $\pi_2(ia)$ of resulting interaction analyses are in the set of instance descriptors $dom_2(gen_{IA}) = \mathcal{I}$. ♣

As each analysis starts with resources allocated for a certain resource-driven process, relevant derived resources are also considered relevant resources of the respective resource-driven processes. Furthermore, it is possible to conduct a second iteration of the analysis by collecting interactions of instances of relevant resources identified after analyzing allocated resources of a resource-driven process. Consequently, interaction collectors will identify additional instances of relevant resources communicating with previously identified instances of relevant resources. Interestingly, these relevant resources can be useful for the future executions of a business process. For example, if a developer identified as a relevant resource updates an external Wiki service with information important for the process execution, this Wiki service becomes relevant, too. Therefore, in certain settings, it makes sense to continue analyzing resources iteratively using the instances of relevant resources identified in the previous iteration to discover relevant resources. Naturally, the likelihood to increase the performance of future executions of business processes using relevant resources identified during later iterations is smaller than relevant resources identified in earlier iterations. Thus, keeping the information regarding the iteration, in which a relevant resource has been identified, is important. Therefore, each interaction analysis keeps the information regarding the iteration in the field *iterationCount*. Moreover, each interaction analysis contains the contents of the interaction for further analysis, such as making certain conclusions about the contributions made by the resource based on the interactions. For example, based on an interaction containing a commit message, it will be possible to conclude that this developer is more likely important for the future executions of the same process, as the developer updated certain parts of the source code. In addition, it will be possible to conclude that a software development capability is possibly a relevant capability for such a business process. Thus, it is possible to discover certain possibly relevant capabilities using capabilities of relevant resources:

Definition 47 (Relevant Capability (informal)). A relevant capability is a capability provided by a relevant resource. ♣

For example, in case a developer was identified, his or her capabilities are relevant capabilities, such as the Java development and test engineering capabilities. Relevant resources and relevant capabilities can have different degrees of relevance based on different factors, such as the contents and types of the interactions collected to identify these resources and capabilities. To quantify the concept of relevance of resources and capabilities with resource-driven processes, relevance relationships are used:

Definition 48 (Relevance Relationship). Let a relevance relationship be $rr \in RR$. Each relevance relationship is a quadruple:

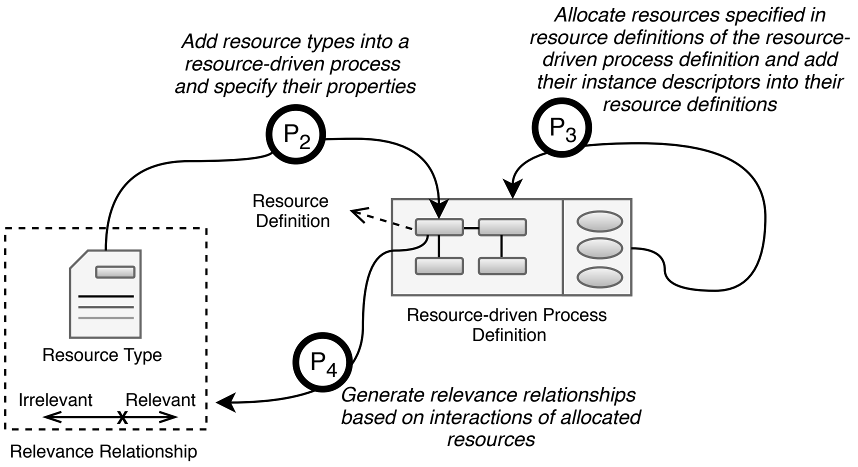
$$RR \subseteq IE \times ID_{od} \times \mathbb{R} \times \wp(ID)$$

$$rr = (ie, id, correlationCoefficient, ID_{rr})$$

Where:

- $\pi_1(rr) = ie$, $ie \in IE$ is an identifiable entity definition specifying a relevance relationship,
- $\pi_2(rr) = id$, $id \in ID_{od}$ is an entity identity referring to a resource type or capability definition of a relevant resource or relevant capability, respectively, in an organizational definition od,
- $\pi_3(rr) = correlationCoefficient, correlationCoefficient \in \mathbb{R}$ is a number quantifying the relevance of a relevant resource or a relevant capability, such that a negative value means negatively relevant, 0 means irrelevant, and a positive value means positively relevant (Definition 55),
- $\pi_4(rr) = ID_{rr}$, $ID_{rr} \subseteq \wp(ID)$ is the set of entity identities referring to interaction analysis and relevance relationships used to derive this relevance relationship. ♣

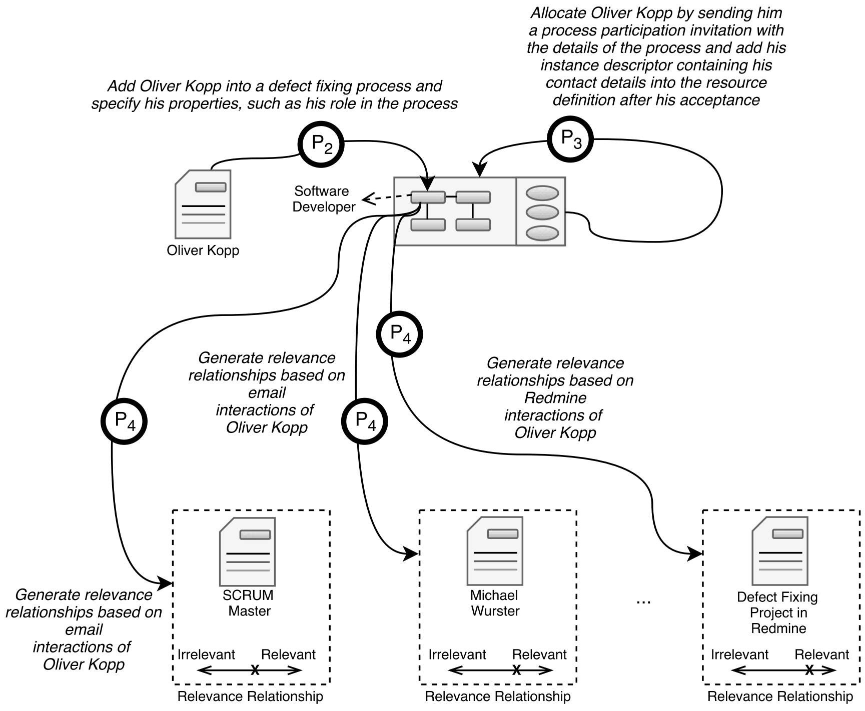
Each relevance relationship is specified with an identifiable entity definition; no two relevance relationships exist with the same entity identity. Moreover, each relevance relationship points to a relevant resource or to a relevant capability. For example, a relevance relationship can point to a resource type representing a certain developer as illustrated in Figure 4.7 or to a certain capability provided by the relevant resource, such as a Java development capability. In addition, each relevance relationship contains a correlation coefficient to specify the degree of relevance of the associated relevant resource or relevant capability. Negative values of the correlation coefficient mean a negative relevance, that is, business experts should not add a relevant resource or a relevant capability into a resource-driven process definition



Add resource types into a resource-driven process and specify their properties
Allocate resources specified in resource definitions of the resource-driven process definition and add their instance descriptors into their resource definitions
P_2
P_3
Resource Definition
Resource Type
Resource-driven Process Definition
P_4
Irrelevant
Relevant
Generate relevance relationships based on interactions of allocated resources
Relevance Relationship

Figure 4.6: A simplified view of resource types, resource definitions, and relevance relationships in P_2, P_3, and P_4.

during P_2, as these resources and capabilities have a negative effect on the process execution. For example, a user spamming a mailing list during the enactment of a business process can be a negatively correlated relevant resource of the business process. The smaller the value of a negative correlation coefficient, the more negative effects a relevant resource or relevant capability has. Moreover, in certain situations, it can be concluded that inclusion of a relevant resource or a relevant capability will probably not change anything. In such cases, a correlation coefficient with the value 0 is used. For example, a user asking a question in the mailing list can be identified as a relevant resource, but he or she will not be considered either negatively correlated or positively correlated. Finally, relevant resources and capabilities creating a positive influence on business process executions are assigned positive correlation coefficients, such as a developer making commitments in a business process. The larger the value of a positive correlation coefficient, the more positive effects a relevant resource or relevant capability has. Figure 4.6 presents a simplified view of representations of



Add Oliver Kopp into a defect fixing process and specify his properties, such as his role in the process
Allocate Oliver Kopp by sending him a process participation invitation with the details of the process and add his instance descriptor containing his contact details into the resource definition after his acceptance
P2
P3
Software Developer
Oliver Kopp
P4
P4
P4
Generate relevance relationships based on email interactions of Oliver Kopp
Generate relevance relationships based on Redmine interactions of Oliver Kopp
Generate relevance relationships based on email interactions of Oliver Kopp
SCRUM Master
Michael Wurster
...
Defect Fixing Project in Redmine
Irrelevant
Relevant
Relevance Relationship

Figure 4.7: A simplified example of resource types, resource definitions, and relevance relationships in P_2, P_3, and P_4.

resources in P_2, P_3, and P_4 of the RPM life cycle. Moreover, Figure 4.7 illustrates a concrete example of this simplified view. During modeling of resource-driven processes (P_2), business experts create resource definitions based on resource types. After initializing resource-driven processes (P_3), instance descriptors of the allocated resources are added to their resource-definitions, as shown in Figure 4.6 and Figure 4.7. Finally, using instance descriptors of resource definitions, relevance relationships are generated during P_4. Generating relevance relationships requires software components called relevance mappers:

Definition 49 (Relevance Mapper (informal)). Relevance mappers are software components capable of generating relevance relationships based on a set of interaction analyses and a set of relevance relationships. ♣

More specifically, the function for this mapping can be specified as follows:

Definition 50 (Relevance Relationship Generator). Let a relevance relationship generator be a function:

$$gen_{RR} : RP \times \wp(RR) \times \wp(IA) \rightarrow \wp(RR)$$

Where:

- $dom_1(gen_{RR}) = RP$ is a set of relevance mappers,
- $dom_2(gen_{RR}) = \wp(RR)$ is the power set of relevance relationships,
- $dom_3(gen_{RR}) = \wp(IA)$ is the power set of interaction analyses,
- $img(gen_{RR}) = \wp(RR)$ is the power set of relevance relationships. ♣

The relevance relationships generator function uses a certain relevance mapper, an existing set of relevance relationships, and a set of interaction analyses. Correlation coefficients of relevance relationships will support the decision-making process of business experts by presenting possibly relevant resources and relevant capabilities during resource-driven process modeling (P_2). For example, a relevance relationship containing a software developer and a correlation coefficient of 10 will recommend the inclusion of this relevant resource depending on the other correlation coefficients of relevance relationships. Moreover, using relevance relationships generated for resource-driven processes, it is possible to generate new resource-driven process definitions automatically, containing the relevant resources and relevant capabilities of the relevant relationships. For example, it is possible to generate a new resource-driven process definition automatically with relevant resources at correlation coefficients higher than 0.8. To automate the generation of new resource-driven process definitions based on a set of relevance relationships and an existing resource-driven process definition

(Figure 4.4), we introduce the concept of resource-driven process definition recommenders:

Definition 51 (Resource-driven Process Definition Recommender (informal)). Resource-driven process definition recommenders are software components using a set of relevance relationships to create a new resource-driven process definition based on a given resource-driven process definition. ♣

Different resource-driven process definition recommenders can rely on different criteria, such as filtering out the relevance relationships below the arithmetical average of the correlation coefficients of all given relevance relationships. A function for generating resource-driven process definitions from resource-driven process definition recommenders, resource-driven process definitions, and sets of relevance relationships is specified as follows:

Definition 52 (Resource-driven Process Definition Generator). Let a resource-driven process definition generator be a function:

$$gen_{\mathcal{P}} : \mathcal{P}R \times \mathcal{P} \times \wp(RR) \rightarrow \mathcal{P}$$

Where:

- $dom_1(gen_{\mathcal{P}}) = \mathcal{P}R$ is the set of resource-driven process definition recommenders,
- $dom_2(gen_{\mathcal{P}}) = \mathcal{P}$ is the set of resource-driven process definitions,
- $dom_3(gen_{\mathcal{P}}) = \wp(RR)$ is the power set of relevance relationships,
- $img(gen_{\mathcal{P}}) = \mathcal{P}$ is the set of resource-driven process definitions. ♣

The function enriches a resource-driven process definition based on the provided relevance relationships using the given resource-driven process definition recommender. The output of the function is the enriched resource-driven process definition. Business experts check this enriched process definition for its appropriateness and adapt it if necessary. Consequently, the automated generation does not avoid the interference of business experts, but aims at reducing the effort spent during the modeling of resource-driven processes (P_2). In the scope of this work, we call generated (i) relevance relationships and (ii) resource-driven process definitions resource-centric recommendations because of being centered on resources and their capabilities. To this end, the adjective resource-driven emphasizes the role of resources in resource-driven processes, whereas the adjective resource-centric emphasizes the focus of the generated recommendations.

Organizations adapt interaction collectors, relevance mappers, and resource-driven process definition recommenders based on their organizational needs during P_1, that is, these are organization-specific software components. Moreover, all these components are registered and orchestrated by a resource-driven process discoverer:

Definition 53 (Resource-driven Process Discoverer (informal)). Resource-driven process discoverers are software components orchestrating interaction collectors, relevance mappers, and resource-driven process definition recommenders to generate relevance relationships and new resource-driven process definitions based on the relevance relationships. ♣

This orchestration of different software components will be detailed in Section 4.3.4. The combination of interaction collectors, relevance mappers, resource-driven process definition recommenders, and resource-driven process discoverers comprises a resource-driven process discovery environment:

Definition 54 (Resource-driven Process Discovery Environment). Let a resource-driven process discovery environment be $de \in DE$. Each resource-driven process discovery environment is a nine-tuple defined as follows:

$$de = (od, po, IC_{de}, RP_{de}, \mathcal{P}R_{de}, gen_{IA}, gen_{RR}, gen_{\mathcal{P}}, \sigma_{ME})$$

Where:

- $\pi_1(de) = od$, $od \in OD$ is an organizational definition containing modeling elements available in the resource-driven process discovery environment (Definition 9),

- $\pi_2(de) = po$, $po \in \mathcal{PO}$ is a resource-driven process discoverer (Definition 53),
- $\pi_3(de) = IC_{de}$, $IC_{de} \subseteq \wp(IC)$ is the set of interaction collectors, such that $1 \leq |IC_{de}|$ (Definition 43),
- $\pi_4(de) = RP_{de}$, $RP_{de} \subseteq \wp(RP)$ is the set of relevance mappers, such that $1 \leq |RP_{de}|$ (Definition 49),
- $\pi_5(de) = \mathcal{P}R_{de}$, $\mathcal{P}R_{de} \subseteq \wp(\mathcal{P}R)$ is the set of resource-driven process definition recommenders (Definition 51),
- $\pi_6(de) = gen_{IA}$ is a function mapping from an interaction collector and an instance descriptor, representing a resource definition to a set of interaction analyses (Definition 46),
- $\pi_7(de) = gen_{RR}$ is a function mapping from a relevance mapper, a set of relevance relationships, and a set of interaction analyses to a set of relevance relationships (Definition 50),
- $\pi_8(de) = gen_{\mathcal{P}}$ is a function mapping from a resource-driven process definition recommender, a resource-driven process definition, and a set of relevance relationships to a resource-driven process definition (Definition 52),
- $\pi_9(de) = \sigma_{ME}$ is a function mapping from an entity identity of a modeling element to the modeling element (Definition 32). ♣

As illustrated in Figure 4.4, each resource-driven process discovery environment is composed of an organizational definition, a resource-driven process discoverer, interaction collectors, relevance mappers, resource-driven process definition recommenders, and functions for generating different recommendations. Moreover, each resource-driven process discovery environment contains at least one interaction collector and relevance mapper, so that a resource-driven process discovery environment is capable of creating relevance relationships. Missing interaction collectors and relevance mappers will result in missing relevance relationships. Furthermore, missing relevance relationships means no recommendations. Interaction collectors, relevance mappers, and resource-driven process definition recommenders are domain-specific software components, meaning that each organization specifies their custom set of interaction collectors, relevance mappers, and resource-driven process definition recommenders based on their needs, as detailed in Section 4.3.1. Next, we detail the calculation of correlation coefficients of relevance relationships.

4.2.3.1 Calculation of the Correlation Coefficient

Correlation coefficients of relevance relationships (Definition 48) provide quantitative means of representing the nature of the relevance between relevant resources or relevant capabilities and resource-driven process definitions. More specifically, a negative value means a relevant resource or a capability has negative effects on a resource-driven process. For example, a team member can be negatively correlated to a business process due to submitting unproductive emails to teammates during a resource-driven process execution. Thus, negative values indicate that business experts should exclude relevant resources or relevant capabilities in corresponding relevance relationships of resource-driven process definitions during the modeling (P_2). In contrast, a positive correlation coefficient indicates that a relevant resource or a relevant capability should be included in a resource-driven process definition. For example, if multiple team members commit to a certain Git repository during a business process enactment, this repository is considered to be positively correlated due to the commit interactions with allocated resources of the business process. Finally, a correlation coefficient with the value 0 means that there is insufficient information to make a conclusion. For example, this situation can occur if a relevant resource is identified for making interactions, resulting in both an increase and decrease in the correlation coefficient of the respective relevance relationship, such as a developer sending spam messages but still contributing to a project. To calculate the correlation coefficient of relevance relationships, we have identified three requirements.

Adjustable Calculation Based on Different Information Sources (Rq_{cc-1}). Calculating correlation coefficients requires considering different kinds of factors, such as types of interactions between instances of analyzed and relevant resources, contents of interactions, and types of analyzed resources. Each of these factors can have different semantic meanings and can result in different correlation coefficients in different organizations. For example, a "commit" interaction can result in a higher relevance (i.e., a larger correlation coefficient) than a "pull" interaction in the context of Git interactions. Thus, there is a need for a means to adjust the calculation of the correlation coefficient based on different information sources available in the corresponding environment, such as contents and types of interactions and existing relevance relationships.

Correlation Coefficient with Memory (Rq_{cc-2}). Multiple interaction analyses can lead to the same relevant resource or capability. Similarly, different relevant resources can lead to the same relevant capability. For example, the same instance of a relevant resource can appear in interaction analyses of a "commit" interaction and a "pull" interaction. Similarly, the relevant resources representing the roles of a software architect and project manager will lead to a "coordination" capability, as both of these need to coordinate team members while executing business processes. Thus, each interaction analysis and each relevance relationship leading to the same relevant resources and capabilities should have an effect on the resulting correlation coefficient.

Considering the Effects of an Iteration Count (Rq_{cc-3}). Interaction collectors initially start gathering interactions of allocated resources of a resource-driven process. Furthermore, interaction collectors can proceed iteratively by collecting interactions of instances of relevant resources identified during the analysis of interactions of allocated resources. Interestingly, these iterations can continue arbitrarily, using the previously identified instances of relevant resources to identify new relevant resources. Naturally, increasing the iteration number will result in a relative smaller increase or decrease in a correlation coefficient than a former iteration with the same characteristics, such as the same type of interaction with the same contents. For example, a developer allocated for a resource-driven process updates a Git repository, which is updated by another developer. Consequently, there is a probability that this second developer is relevant for the resource-driven process specified, but this probability is smaller than the probability of the first developer being relevant for the future executions of the respective business process. Similarly, a relevant capability provided by a relevant resource discovered in a second iteration is probably less relevant than a capability discovered during the first iteration. During the calculation of correlation coefficients, there should be a means of including the effects of relevant resources identified during different iterations. Based on these requirements, we created the following equation.

Definition 55 (Correlation Coefficient Calculator). Let a correlation coefficient calculator be a function:

$$gen_{CC} : ID_{od} \times \wp(IA \cup RR) \rightarrow \mathbb{R}$$

$$gen_{CC}(id_{rc}, IA \cup RR) = \sum_{iarr \in IA \cup RR} \frac{rFactor(id_{rc}, iarr)}{iFactor(iarr)}$$

Where:

- $dom_1(gen_{CC}) = ID_{od}$, $id_{rc} \in ID_{od}$ is a set of entity identities indicating a resource type or a capability definition,
- $dom_2(gen_{CC}) = \wp(IA \cup RR)$ is the power set of the union set of interaction analyses and relevance relationships used to calculate the correlation coefficient,

- $img(gen_{CC}) = \mathbb{R}$ is the resulting correlation coefficient,
- $rFactor : ID_{od} \times IA \cup RR \rightarrow \mathbb{R}$ is a function mapping an entity identity id_{rc} referring to (i) a relevant resource or relevant capability and (ii) an interaction analysis or a relevance relationship to a relevance factor,
- $iFactor : IA \cup RR \rightarrow \mathbb{Z}^+$ is a function mapping an interaction analysis or a relevance relationship to an iteration factor. ♣

The gen_{CC} function maps an entity identity referring to a resource type or a capability definition and the union set of interaction analyses and relevance relationships to a correlation coefficient. More specifically, the effect of each interaction analysis and relevance relationship is given as input calculated and added during the calculation of a corresponding correlation coefficient. For example, the function generates a correlation coefficient for an entity identity referring to the resource type "software developer" or to the capability definition of a "project management" capability. Therefore, the gen_{CC} function generates a specific value for each given interaction analysis and each relevance relationship using the *rFactor* function and *iFactor* function. Thereafter, the function sums all the results up to find an overall correlation coefficient of the considered resource type or capability definition. For example, for a software developer provided with 15 interaction analyses of Git commit interactions to gen_{CC}, the function will create a specific value for each interaction analysis using *rFactor* and *iFactor*. Hereafter, all the these individual results will be summed up to find the resulting correlation coefficient representing the degree of the relevance of the software developer.

The *rFactor* stands for the relevance factor of an interaction analysis or relevance relationship for the given relevant resource or relevant capability. For example, the *rFactor* function will return a positive value for an entity identity representing a software developer and an interaction analysis of a Git commit done by the software developer. Similarly, a software development capability and a relevance relationship containing a software developer given to *rFactor* will result in a positive value. Thus, the software developer increases the relevance of the given capability. In contrast, the *rFactor* function will return a negative value for an entity identity representing a software developer and an interaction analysis of a spam email sent by the software developer.

Each relevance factor is divided by an integer calculated using *iFactor*, resulting in an iteration factor. Iteration factors are used to distinguish interaction analyses gathered during different iterations, as explained in Rq_{cc-3}. For example, interaction analyses gathered for a software developer identified by analyzing a Git repository defined in a resource-driven process will return a larger result from *iFactor* in comparison to an interaction analysis gathered during the analysis of the Git repository meaning that resources interacting more directly with resources allocated for resource-driven processes are more relevant. If relevance relationships are input parameters to *iFactor*, the function can use relevant resources included in the relevance relationships to calculate the result of *iFactor*. For example, a relevance relationship of a software tester and a software developer with correlation coefficients 8 and 5 are given to *iFactor* during the calculation of a correlation coefficient of a software testing capability. Considering that both software developer and tester provide the software testing capability, the relevance relationship with the lower correlation coefficient (software developer) will lead to a larger result of *iFactor* in comparison to the relevance relationship with the larger correlation coefficient (software tester). Thus, the capability of a less relevant resource is less relevant, too. With an increasing iteration factor, the resulting correlation coefficient will decrease. After dividing each relevance factor through an iteration factor for each interaction analysis and relevance relationship, the resulting values are summed to calculate the final correlation coefficient, specifying the degree of relevance of a resource type or capability definition.

Technical experts design and implement *rFactor* and *iFactor* functions during P_1 inside of relevance mappers. For example, they assign mappings to constant values from different types of interactions available in the respective organization, such as a Git commit interaction, Git branch interaction, and Git merge interaction during the design of *rFactor*. Hereafter, technical experts can create a database table containing these mappings between interactions and their resulting values. During the generation of relevance relationships, these functions are executed automatically. For example, an *rFactor* function can map from a certain type of interaction to a specific value using the database table defined by technical experts. Similarly, an *iFactor* function can use the correlation coefficient of a relevant resource in a relevance relationship or *iterationCount* in an interaction analysis to calculate an iteration factor, such as creating a larger result for a relevance relationship with a correlation coefficient of 15 with respect to a relevance relationship with a correlation coefficient of 30.

The *rFactor* function provides adjustment effects of different factors, such as the interaction types, interaction contents, and relevance relationships. Thus, the equation is expected to fulfill Rq_{cc-1}, i.e., technical experts designing and implementing the function behave correctly. Moreover, by relying on all different interaction analyses and relevance relationships during the calculation of correlation coefficients, the resulting correlation coefficients include effects of all interaction analyses and relevance relationships. Consequently, the equation meets the requirement Rq_{cc-2}. The iteration factor *iFactor* enables including the effects of the interaction analyses collected during different iterations. Furthermore, relevant capabilities identified using relevant resources with a larger *iFactor* will also have a larger *iFactor*, resulting in a lower correlation coefficient. As a result, the equation meets the requirement Rq_{cc-3}. In the following sections, we describe each phase in detail, using the concepts introduced in the conceptual framework.

4.3 Detailed Explanation of the Resource-driven Process Management Life Cycle

This section presents a detailed explanation of the Resource-driven Process Management (RPM) life cycle using the concepts introduced in Section 4.2. Next, we explain the preparation phase (P_1) of the RPM life cycle.

4.3.1 Preparing the Resource-driven Process Environment (P_1) using the Conceptual Framework

The first phase of the RPM life cycle aims at adapting (i) a resource-driven process modeling environment (Definition 33), (ii) a resource-driven process execution environment (Definition 42), and (iii) a resource-driven process discovery environment (Definition 54) of an organization to the needs of business processes. As a result of this phase, organizations enable (i) visibility of available resources and their relationships in modeling environments of resource-driven processes, (ii) automated allocation of interrelated resources of resource-driven process definitions, and (iii) automated generation of recommendations for business experts, creating resource-driven process definitions based on allocated resources. During this adaptation, business and technical experts adjust domain-specific components presented in Section 4.2, such as domain-specific information providers, domain-specific operation providers, interaction collectors, relevance mappers, and resource-driven process definition recommenders. This phase involves two different roles to conduct the different steps involved. The first role is business experts with knowledge about organizational business processes. Their knowledge comprises resources and their relevant interrelationships for executing the business processes of their organization. In addition, the second role is technical experts with knowledge about IT aspects of these resources and their relationships. Technical experts are familiar with web services capable of (i) representing resources and their relationships, (ii) automating their allocation, and (iii) storing event logs containing interactions among resources, such as Docker [Tur14], Elgg [Sha08], GitHub [BB14], and OpenTOSCA [BBH+13]. During this phase, business experts and technical experts execute different sets of steps, as presented in Figure 4.8. Each step is abbreviated with the letter "I," as the phase symbolizes the initial steps. In the following, we describe each of these steps separately.

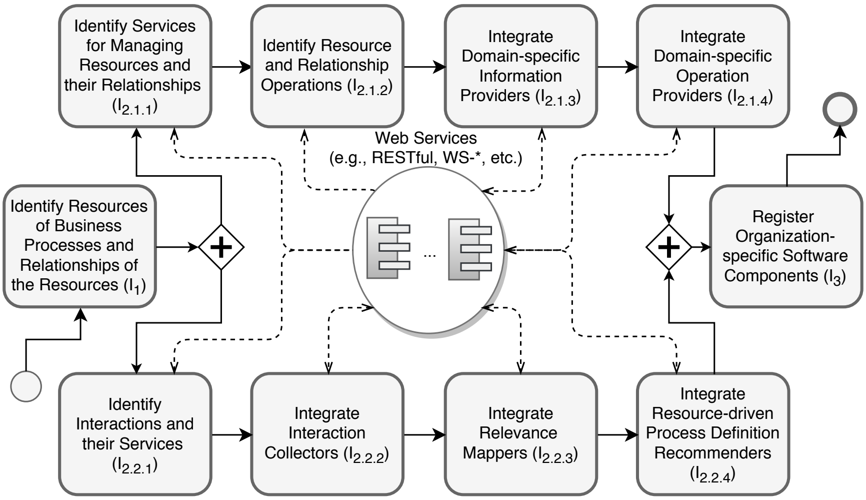

Identify Services for Managing Resources and their Relationships ($I_{2.1.1}$)
Identify Resource and Relationship Operations ($I_{2.1.2}$)
Integrate Domain-specific Information Providers ($I_{2.1.3}$)
Integrate Domain-specific Operation Providers ($I_{2.1.4}$)
Web Services (e.g., RESTful, WS-*, etc.)
Identify Resources of Business Processes and Relationships of the Resources (I_1)
Register Organization-specific Software Components (I_3)
Identify Interactions and their Services ($I_{2.2.1}$)
Integrate Interaction Collectors ($I_{2.2.2}$)
Integrate Relevance Mappers ($I_{2.2.3}$)
Integrate Resource-driven Process Definition Recommenders ($I_{2.2.4}$)

Figure 4.8: Steps for preparing resource-driven process environments in organizations.

4.3.1.1 Identify Resources of Resource-driven Processes and their Relationships (I_1)

Business processes of different organizations typically contain variant sets of resources to achieve their organizational goals. For example, an organization producing mobile devices will have business processes involving a software expert, a version control system, hardware experts, and a knowledge base. In contrast, a business process in the car manufacturing domain will have mechanical engineers and computer-aided design tools. Moreover, typically there are different relationships between different resources, such as the software experts who "use" the integrated development environment and the manager who has "admin" privileges of the knowledge base. In the first step of P_1, business experts identify resources that contribute to the accomplishment of goals of business processes of their organization. To identify these, they rely on their experience and conduct interviews with the actors of business processes, e.g., they identify software experts, Git repositories, and Wiki services in the case of the business process aimed at resolving a mobile device defect, as these are critical for achieving the goal of resolving the defect. In addition, they analyze relationships of resources enabling means of coherent collaboration in the teams, such as a “managing” relationship between a Wiki service and actors to assign appropriate rights to the desired actors. As a result of conducting I_1, business experts identify a list of resources and relationships. As the identified resources and relationships grant a basis for the followings steps of this phase and, consequently, the following phases of the RPM life cycle, I_1 should be conducted with great care.

As illustrated in Figure 4.8, the following steps are executed in parallel. The steps ($I_{2.1.1}$ – $I_{2.1.4}$) involved in the upper branch of Figure 4.8 target (i) preparing a resource-driven process modeling environment and (ii) resource-driven process execution by integrating a set of domain-specific information and operation providers. Integrating domain-specific software components indicates either reusing or developing a set of web services. Moreover, the steps ($I_{2.2.1}$ – $I_{2.2.4}$) involved in the lower branch of Figure 4.8 aim at preparing the resource-driven process discovery environment. Next, we start by explaining the steps of the upper branch, although these steps can be executed in parallel.

4.3.1.2 Preparing the Resource-driven Process Modeling and Execution Environment ($I_{2.1}$)

After identifying resources and relationships during I_1, technical experts analyze alternative ways of (i) making these resources and relationships available in modeling environments of resource-driven processes and (ii) automating the allocation of the identified interrelated resources. To minimize the effort of creating new domain-specific information providers and domain-specific operation providers, technical experts analyze existing IT services that (i) already present the identified resources and relationships in their domain-specific format or (ii) automate the allocation of the allocated resources ($I_{2.1.1}$). For example, in the case of human resources, social networks are capable of presenting human resources of organizations. Similarly, for IT resources, OpenTOSCA container or Docker provide a means of listing available resources and automating their allocation.

After identifying existing IT services capable of managing resources and their relationships, technical experts proceed with $I_{2.1.2}$, which aims at identifying operations needed for automating the allocation of interrelated resources specified in resource-driven process definitions. Establishing different types of relationships can require executing different operations on different resources, such as adding an admin to a Wiki service in the case of a "managing" relationship between a software developer and the Wiki service. Therefore, during $I_{2.1.2}$, technical experts analyze required operations to automate the allocation of interrelated resources.

During $I_{2.1.3}$ and $I_{2.1.4}$, technical experts start integrating domain-specific information and operation providers capable of (i) making the identified resources and their relationships available in resource-driven process modeling environments and (ii) automating the allocation of the interrelated resources by providing an appropriate list of operation endpoints identified in $I_{2.1.2}$. During the integration, technical experts first determine if desired domain-specific information and operation providers already exist to reuse these. If they do not exist, they develop new domain-specific information and operation providers. Therefore, technical experts reuse the IT services identified in $I_{2.1.1}$ during the development of domain-specific information providers and domain-specific operation providers to reduce the implementation effort, if applicable. For example, technical experts can reuse the open-source social network service Elgg in domain-specific information providing human resources. Similarly, they can reuse the IT resource management functionalities provided by the OpenTOSCA container to deliver operation endpoints of a domain-specific operation providing IT resources.

If no such reusable IT services have been identified, technical experts need to develop business logic that is capable of (i) managing organizational resources and their relationships and (ii) automating the allocation of the interrelated resources from scratch. For example, organizations can have social networks but may not have an automated web service capable of communicating with their human resources over the social network to allocate them to business processes. In this case, technical experts can develop a web service in the form of an automated business process, accomplishing the allocation and deallocation of human resources by communicating with them over desired social networks of respective organizations. They integrate this web service to a domain-specific operation provider capable of delivering the operation endpoint of the web service to the execution environments of resource-driven processes. Next, we describe detailed steps of $I_{2.2}$.

4.3.1.3 Preparing the Resource-driven Process Discovery Environment ($I_{2.2}$)

Discovery environments of resource-driven processes (Definition 54) present business experts with relevant resources, relevant capabilities, and new resource-driven process definitions to support them when modeling resource-driven processes. To generate these relevant resources and relevant capabilities, interaction collectors gather interaction analyses of (i) allocated resources for resource-driven processes and (ii) identified instances of relevant resources. For example, an interaction collector of Git repositories can investigate a Git repository included in a resource-driven process to identify relevant resources, such as relevant software developers. As each organization typically relies on different types of resources to achieve their goals, types of interactions that need to be collected for identifying relevant resources and capabilities can change. For example, if an organization relies on Subversion (SVN) instead of Git, interactions in their business processes will have SVN interactions. Consequently, interaction collectors should be capable of analyzing SVN repositories to deliver interaction analyses. Therefore, during $I_{2.2.1}$, business experts and technical experts identify interactions, indicating relevant resources and relevant capabilities. Interestingly, interactions conducted during the achievement of organizational goals will typically imply relevant resources and capabilities. Thus, business experts and technical experts concentrate on different types of interactions of the resources identified during I_1 in the context of goals. For example, if a software development company relies on Git as a version control system, they identify different kinds of interactions happening with Git repositories, such as a Git commit interaction, a Git branch interaction, and a Git merge interaction. Similarly, if the organization relies on a Wiki service as its knowledge base, they identify interactions, such as reading or editing content stored in the respective Wiki service. Moreover, during the identification of interactions, the privacy of the organizational members should also be considered. Identifying these interactions is followed by the identification of services storing these interactions. For example, a Git server stores all information regarding changes in the stored source code. Similarly, a Wiki service, such as MediaWiki, can provide access to logs regarding reads and writes on the information stored in the corresponding Wiki service.

After identifying interactions and their services, technical experts reuse or develop interaction collectors capable of gathering the desired types of interactions ($I_{2.2.2}$). They either reuse existing interaction collectors or develop new ones from scratch. If they need to develop new ones from scratch, they rely on the services identified during $I_{2.2.1}$ to collect the identified interactions. For example, an interaction collector of Git repositories can collect interactions using a web service providing these interactions. Next, technical experts reuse or develop relevance mappers capable of deriving relevance relationships from the interactions delivered by the interaction collectors integrated previously ($I_{2.2.3}$). To assign different correlation coefficients for different types of interactions, technical experts consult business experts, as business experts are assumed to be more familiar with the implications of different types of interactions. For example, business experts indicate that a commit interaction means a higher relevance than a pull interaction in the context of Git interactions. As a result of integrating relevance mappers, discovery environments of resource-driven processes can present relevant resources and relevant capabilities to support business experts creating resource-driven process definitions. To automate the generation of recommendation models using a set of relevance relationships, technical experts reuse or develop resource-driven process definition recommenders ($I_{2.2.4}$). As shown in Figure 4.1, this phase is executed at the beginning of each cycle because changing technologies and organizational requirements can lead to the inclusion of new resources. Next, we describe the registration step of different domain-specific software components developed so far.

4.3.1.4 Register Organization-specific Software Components (I_3)

After the integration steps $I_{2.1.3}$, $I_{2.1.4}$, $I_{2.2.2}$, $I_{2.2.3}$, and $I_{2.2.4}$, technical experts register domain-specific information providers, domain-specific operation providers, interaction collectors, relevance mappers, and resource-driven process definition recommenders to finalize the integration. Registering interaction collectors, relevance mappers, and resource-driven process definition recommenders at resource-driven process discoverers results in an update in the available set of only these components. In contrast, registering domain-specific information providers and domain-specific operation providers causes additional business logic to be executed. The main purpose of this additional business logic is adding operation endpoints supporting certain resource and relationship types. For example, if there are two domain-specific operation providers with operation endpoints supporting a Wiki service, registering these two domain-specific operation providers will result in an updated resource type of the Wiki service containing these two operation endpoints. Furthermore, this additional business logic for registering domain-specific operation providers and domain-specific information providers can be described algorithmically, unlike the other steps presented in P_1. Thus, we show two algorithms (Algorithm 1 and Algorithm 2) representing the business logic required for the registration in the following paragraphs.

We present a registration algorithm of domain-specific information providers in Algorithm 1. The defined algorithms follow the conventions from [CLRS09], for example, the "entity identity" attribute of a "resource definition" is accessed using $id[ie[ad[sd]]]$. To this end, sd refers to a resource definition and its initializable entity definition (ad) is accessed using the notation $ad[sd]$. Furthermore, objects are passed with their references. Consequently, if the function updates an object passed by the caller, it is visible to the caller.

The registration algorithm has three input parameters: (i) an organizational definition, at which a domain-specific information provider will

Algorithm 1 Register a domain-specific information provider.

```
1: procedure REGISTERDOMAINSPECIFICINFOPROVIDER($od, dip, DOP$)
2:     for all $sr_{dip} \in \mathcal{S}_{dip}[dip] \cup \mathcal{R}_{dip}[dip]$ do ▷ Iterate through resource types and relationship types of the domain-specific information provider
3:         for all $dop \in DOP$ do ▷ Iterate through domain-specific operation providers
4:             for all $oeb_{dop} \in OEB_{dop}[dop]$ do ▷ Iterate through all operation bindings of each domain-specific operation provider
5:                 if $id$ of $sr_{dip}$ is in $ID_{sr}[oeb_{dop}]$ then ▷ Does the operation binding support the resource or relationship type?
6:                     $OE_{sr}[sr_{dip}] \leftarrow OE_{sr}[sr_{dip}] \cup \{oe_{dop}[oeb_{dop}]\}$
7:                 end if
8:             end for
9:         end for
10:        if $sr_{dip}$ is resource type then
11:            $\mathcal{S}_{od}[od] \leftarrow \mathcal{S}_{od}[od] \cup \{sr_{dip}\}$ ▷ Add new type to the organizational definition
12:        else $sr_{dip}$ is a relationship type
13:            $\mathcal{R}_{od}[od] \leftarrow \mathcal{R}_{od}[od] \cup \{sr_{dip}\}$ ▷ Add new type to the organizational definition
14:        end if
15:    end for
16: end procedure
```

be registered, (ii) a domain-specific information provider, which will be registered, and (iii) a set of domain-specific operation providers available in the respective organization. As shown at line 2, the registration algorithm of domain-specific information providers iterates through resource and relationship types of the domain-specific information provider given as the second input parameter. Moreover, at line 3, the algorithm iterates through the given domain-specific operation providers for each resource or relationship type. As each domain-specific operation provider is composed of multiple operation bindings, a further iteration is used to control each of these operation bindings at line 4. For each operation binding it is checked, if the operation endpoint associated with the operation binding supports the new resource or relationship type provided by the given domain-specific information provider at line 5. If the operation endpoint is supported, it is added to the set of operation bindings of the resource or relationship types being considered at line 6. After adding operation endpoints, each resource and relationship type contained in the given domain-specific information provider is added to the set of resource and relationship types of the organizational definition at lines 11 and 13. Consequently, each resource and relationship type provided by each domain-specific information provider is associated with available operation endpoints supporting these types.

The operation for deregistration of each domain-specific information provider is straightforward and results in the removal of (i) each resource type and relationship type and (ii) resource definition and relationship definition referring to the removed types. The presented algorithm is not idempotent and changes the state of organizational definitions. Thus, it should not be called repeatedly with the same parameters.

Algorithm 2 Register a domain-specific operation provider.

```
1: procedure REGISTERDOMAINSPECIFICOPERATIONPROVIDER($od, dop$)
2:     for all $oeb_{dop} \in OEB_{dop}[dop]$ do ▷ Iterate through available operation bindings of the given domain-specific operation provider
3:         for all $sr_{od} \in \mathcal{R}_{od}[od] \cup \mathcal{S}_{od}[od]$ do ▷ Iterate through available resource and relationship types of the given organizational definition
4:             if $id$ of $sr_{od}$ is in $ID_{sr}[oeb_{dop}]$ then ▷ Add operation endpoint of the operation binding to the resource or relationship type if it supports the definition
5:                 $OE_{sr}[sr_{od}] \leftarrow OE_{sr}[sr_{od}] \cup \{oe_{dop}[oeb_{dop}]\}$
6:             end if
7:         end for
8:     end for
9: end procedure
```

Similarly, registering a domain-specific operation provider requires each resource or relationship type to be checked and updated if a new domain-specific operation provider has an operation endpoint supporting the respective resource or relationship type. Algorithm 2 illustrates the necessary business logic for registering domain-specific operation providers. Each registration operation of domain-specific operation providers takes two input parameters: (i) an organizational definition, at which a domain-specific operation provider will be registered and (ii) a domain-specific operation provider, which will be registered. First, the algorithm iterates through all available operation bindings in the new domain-specific operation provider, as stated at line 2. For each operation binding, every existing resource and relationship type in the organizational definition is checked regarding if the operation binding supports the given resource and relationship type. If the resource and relationship types are supported (line 4), the corresponding operation endpoint is added to the set of operation endpoints specified under the respective resource and relationship types at line 5.

Deregistering domain-specific operation providers requires a similar business logic, in the sense that it iterates through all resource and relationship types for each operation binding included in the domain-specific operation providers. The main difference between deregistering and registering a domain-specific operation provider is that each operation endpoint of the domain-specific operation provider supporting resource and relationship types will be removed from the set of resource and relationship types instead of being added. Consequently, the union operation at line 5 of Algorithm 2 will be replaced with a complementary operation ($\setminus$).

Registering interaction collectors, relevance mappers, and resource-driven process definition recommenders requires updating corresponding sets of the respective resource-driven process discovery environment and no further business logic. Thus, upon registration, corresponding sets of resource-driven process discovery environments contain the respective domain-specific software components. Similarly, during deregistration, these are removed from the corresponding sets of resource-driven process discovery environments. After the completion of registration, resource and relationship types are

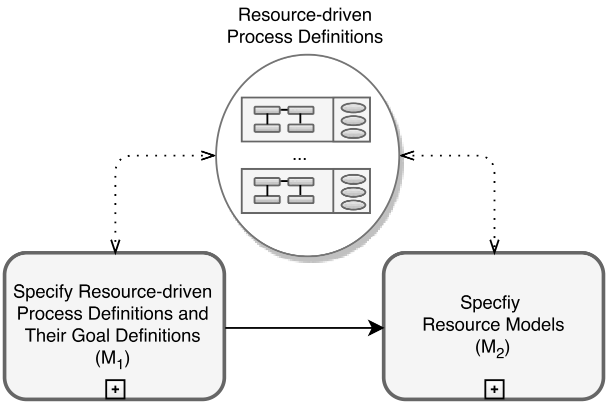

Resource-driven Process Definitions
...
Specify Resource-driven Process Definitions and Their Goal Definitions (M_1)
Specfiy Resource Models (M_2)
+
+

Figure 4.9: Parts of modeling resource-driven processes (P_2).

available in resource-driven process modeling environments. Consequently, business experts can create resource-driven process definitions, which will be detailed in the next section Section 4.3.2. Moreover, after registering domain-specific operation providers, each resource and relationship type contains its supporting operation endpoints for automating the allocation and deallocation of resources for automating the establishment and release of relationships, as discussed in Section 4.3.3.

The computational complexity of these algorithms is not a concern of organizations, as these algorithms are not executed repeatedly and are executed during preparation phases, not during the enactments of resource-driven processes. Finally, using registered interaction collectors, relevance mappers, and resource-driven process definition recommenders, it is possible to create resource-driven recommendations including relevant resources, relevant capabilities, and new resource-driven process definitions generated based on these relevant resources and capabilities, as explained in Section 4.3.4.

4.3.2 Model Resource-driven Processes (P_2) Using the Conceptual Framework

This phase involves steps for creating resource-driven process definitions using resource and relationship types made available in resource-driven process modeling environments during P_1. Furthermore, the presented modeling approach is our Contribution 2. The modeling phase comprises two parts, as shown in Figure 4.9. Executing the first, part M_1, results in resource-driven process definitions associated with goal definitions as detailed in Section 4.3.2.1. During part M_2, business experts detail the resource-driven process definitions created during M_1 with interrelated resources capable of achieving specified goals for the resource-driven process definition. During M_1 and M_2, business experts create modeling elements of Redo using a resource-driven process modeling environment (Definition 33).

Creating resource-driven process definitions requires business experts with knowledge of organizational goals and organizational capabilities required for the goals. Moreover, they need to be aware of (i) different types of resources providing organizational capabilities and (ii) capabilities forming more complex capabilities needed for achieving organizational goals. For example, business experts should know that a goal of resolving a mobile device defect will require capabilities, such as a software development capability, a software developer capability, and a software tester capability. To this end, a software development capability refers to a development environment providing the capability, whereas a software developer capability refers to a software developer. In addition, they should be aware that a software development capability is provided by a Java development environment and a .NET development environment.

During the creation of resource-driven process definitions, they consult actors of business processes, when necessary. Furthermore, resource-driven process definitions can be created (i) proactively and (ii) reactively. To this end, creating the definitions proactively will mean specifying resource-driven process definitions as a precaution, such as a resource-driven process aimed at managing a crisis in an organization. In contrast, creating the definitions reactively will result in the specification of just-in-time resource-driven processes, such as creating a resource-driven process definition for fixing a mobile device defect upon the receipt of a defect report. Both proactively and reactively created resource-driven process definitions can be reused if the corresponding process is repeatedly executed.

During creating modeling elements, business experts exploit resource-driven process modeling tools (Definition 31). Specifically, business experts firstly control the existence of desired capability, context, goal, or resource-driven process definitions (US_1). When such a match exists, they either reuse this or create an updated version of the match (US_2 and US_3). Otherwise, business experts create new capability, context, goal, or resource-driven process definitions from scratch (US_3). Unneeded capability, context, goal, or resource-driven process definitions can be removed any time (US_4). Next, we explain the two parts of the modeling phase in detail. Please note that in the following section, we do not detail the creation of entity definitions (Definition 13) of different modeling elements of Redo, as these can vary between different organizations. For example, a manufacturing company can specify context definitions using machine-processable expressions, whereas an IT company can use human-readable documents. Consequently, specifying entity definitions is accustomed to organizations themselves and are out of the scope of this work.

4.3.2.1 Specify Resource-Driven Process Definitions and Their Goal Definitions (M_1)

Business processes aim at accomplishing one or more goals, such as a goal of investigating and fixing a mobile device defect. These goals are not fixed, but business experts know a set of goals during the creation of a resource-driven process definition. In addition, M_1 focuses on the goals of resource-driven processes. More specifically, the objective of M_1 is specifying a resource-driven process definition associated with its goal definitions for further modifications during M_2. Therefore, business experts examine goals to be reached for completing the business process being modeled during $M_{1.1}$, as

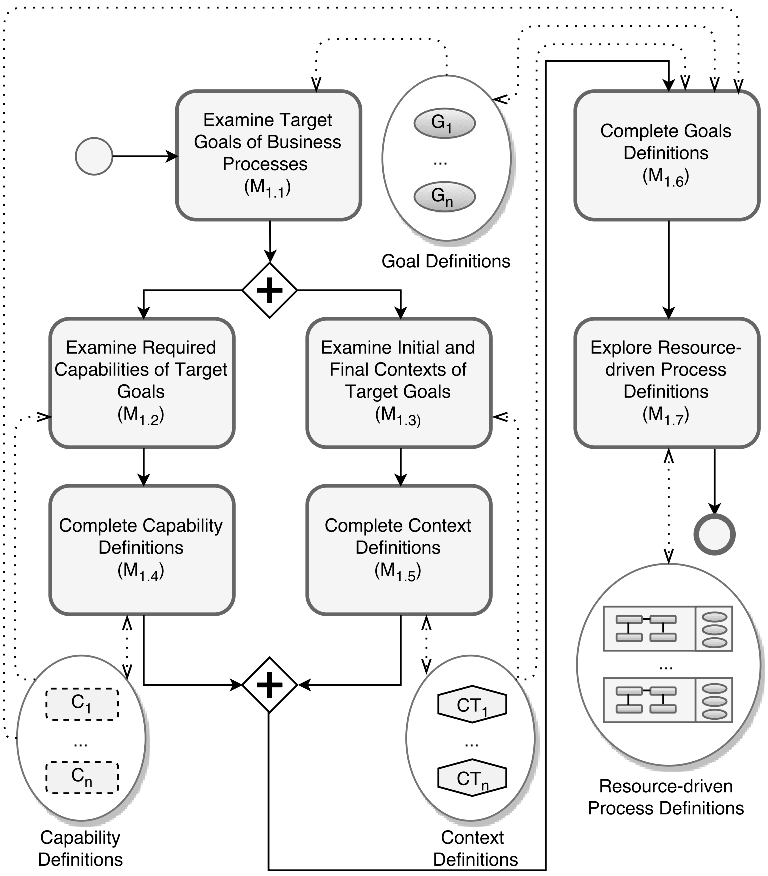

Examine Target Goals of Business Processes ($M_{1.1}$)
G_1
...
G_n
Goal Definitions
Complete Goals Definitions ($M_{1.6}$)
Examine Required Capabilities of Target Goals ($M_{1.2}$)
Examine Initial and Final Contexts of Target Goals ($M_{1.3}$)
Explore Resource-driven Process Definitions ($M_{1.7}$)
Complete Capability Definitions ($M_{1.4}$)
Complete Context Definitions ($M_{1.5}$)
C_1
...
C_n
Capability Definitions
CT_1
...
CT_n
Context Definitions
...
Resource-driven Process Definitions

Figure 4.10: Steps for modeling resource-driven process definitions and their goal definitions.

illustrated in Figure 4.10. For example, a business process that is executed during maintenance of a mobile device can aim at fixing a defect, resolving security issues, and improving the performance of the mobile device. Business experts proceed with either $M_{1.2}$ or $M_{1.3}$. Consequently, the numbering of the steps does not imply any ordering between the steps presented but are just identifiers. Achieving organizational goals requires abilities to perform certain productive tasks (i.e., organizational capabilities). Therefore, business experts examine the capabilities required to accomplish goals identified during $M_{1.1}$. For example, business experts conclude that achieving a goal of fixing a mobile device defect will require certain capabilities, such as a "software developer" capability and a "software tester" capability.

Redo uses capability definitions (Definition 22) to define organizational capabilities. Moreover, each such capability definition is managed using a resource-driven process modeling tool (Definition 31). Therefore, business experts check resource-driven process modeling environments for the existence of desired capability definitions during $M_{1.4}$. If capability definitions representing the identified capabilities do not exist, they create new capability definitions by specifying entity definitions, entity identity, and associated resource types or capability definitions. For example, business experts can create a missing capability definition of a "software development" capability associated with (i) an HTML 5 application providing an integrated development environment and (ii) a virtual machine containing a .NET framework and Microsoft Visual Studio. Furthermore, they can create another capability definition representing a "management" capability associated with a "project manager" capability and a "project management" capability.

Organizational goals represent the intent of moving from a certain organizational context to another context. For example, a goal of fixing a mobile device defect will have the presence of a defect as its initial context due to achieving the goal upon the receipt of defect reports. Moreover, the final context of a goal of fixing a mobile device defect will be a functioning mobile device. Consequently, each organizational goal is associated with an initial context representing the start context before achieving a goal and a final context representing the final context after achieving a goal. Thus, business experts examine initial contexts, under which organizations start accomplishing goals of business processes modeled during $M_{1.3}$, as shown in Figure 4.10. After classifying initial contexts of goals, business experts inspect outcomes of accomplishing goals (final contexts of goals), such as a functioning mobile device without a reported defect after achieving the goal of investigating and fixing the defect. Redo uses context definitions (Definition 27) to describe the initial and final contexts identified during $M_{1.3}$. Moreover, each context definition is managed using resource-driven process modeling environments of organizations. Based on identified initial contexts and final contexts, business experts explore available context definitions addressing these identified initial and final contexts using resource-driven process modeling environments. If such context definitions do not exist, business experts define new context definitions representing the identified initial contexts and final contexts of goals during $M_{1.5}$ using resource-driven process modeling environments.

Resource-driven process modeling environments of organizations can already contain a set of goal definitions. Such existing goal definitions provide a knowledge base for creating goal definitions of a business process aimed at the goals identified during $M_{1.1}$. For example, a goal definition describing a goal of fixing a product can provide the basis for a goal definition of a business process aimed at fixing a mobile device defect. Therefore, in $M_{1.6}$, business experts proceed with checking available goal definitions describing goals identified during $M_{1.1}$ and addressing initial contexts and final contexts identified during $M_{1.3}$. If business experts can detect such goal definitions, they investigate the completeness of these by comparing associated capability definitions of goal definitions with the previously identified capabilities required for goals during $M_{1.2}$. If associated capability definitions of a goal definition do not represent all capabilities required for the goal, business experts associate these missing capability definitions with the corresponding goal definitions or create a new goal definition containing all capability definitions during $M_{1.6}$. For example, if they discover an existing goal definition describing a goal of fixing a mobile device defect without an associated software tester capability, they associate this capability. If no goal definition is available, business experts create goal definitions from scratch by adding entity definitions and entity identities and by associating capability and context definitions representing capabilities identified during $M_{1.2}$ and initial and final contexts identified during $M_{1.3}$, as illustrated in Figure 4.10.

After completing goal definitions describing target goals of the business process being modeled, business experts proceed with either (i) reusing an existing resource-driven process definition or (ii) creating a new resource-driven process definition during $M_{1.7}$. Therefore, business experts explore available resource-driven process definitions associated with the goal definitions specified during $M_{1.6}$. If they detect a resource-driven process definition aimed at the same set of goals, they can reuse the complete resource-driven process definition during $M_{1.7}$, as shown in Figure 4.10. For example, business experts can identify a resource-driven process aiming at fixing a product. Thereafter, they can reuse this resource-driven process definition instead of creating a new one from scratch. If business experts cannot reuse resource-driven process definitions completely, they reuse the information available in the partially matching resource-driven process definitions, aimed at a similar set of goals. For example, they analyze resource and relationship types of the partially matching resource-driven process definitions, such as a Wiki service, Git repository, and project manager, and consider adding these during M_2 to the resource model of the new resource-driven process definitions. In the absence of reusable resource-driven process definitions, business experts create new resource-driven process definitions by specifying entity definitions, entity identity, and associating goal definitions, addressing the goals identified during $M_{1.1}$. As a result of this step, resource-driven process definitions associated with the goals of the modeled business process exist, as shown in Figure 4.10. Next, business experts specify interrelated resources needed for achieving these goals, as explained in the following section.

4.3.2.2 Define Resource Models of the Resource-Driven Process Definitions (M_2)

Business experts document interrelated resources of business processes in resource-driven process definitions. During M_2, the documentation of resources and their relationship for achieving goals of business processes takes place. Therefore, business experts create or adapt resource models representing graphs of interrelated resources capable of achieving goals of business

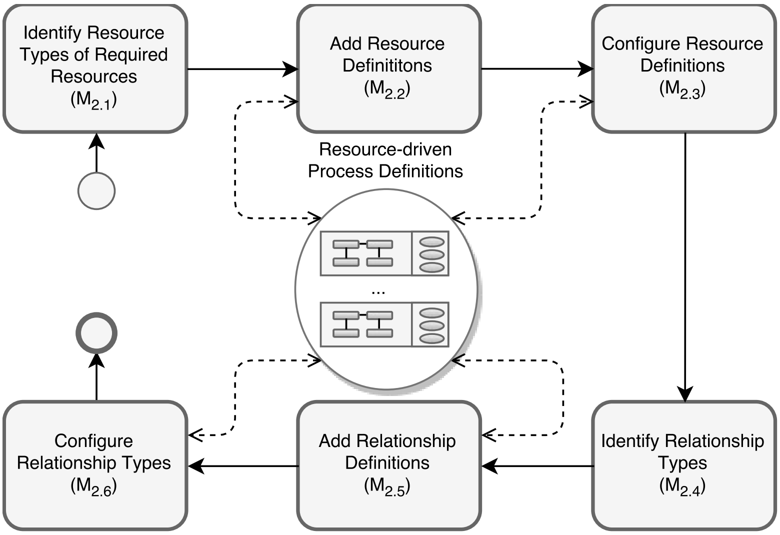

Identify Resource Types of Required Resources ($M_{2.1}$)
Add Resource Definititons ($M_{2.2}$)
Configure Resource Definitions ($M_{2.3}$)
Resource-driven Process Definitions
...
Configure Relationship Types ($M_{2.6}$)
Add Relationship Definitions ($M_{2.5}$)
Identify Relationship Types ($M_{2.4}$)

Figure 4.11: Steps of specifying resource models of resource-driven process definitions.

processes. They create a new resource model if a new resource-driven process definition is created during $M_{1.7}$. In contrast, if business experts reuse an existing resource-driven process definition identified during $M_{1.7}$, they adapt resource models of the reused resource-driven process definition to their current needs. During the creation of resource models of resource-driven process definitions, goal definitions and their associated capability definitions guide business experts. Capability definitions refer to (i) resource types providing the specified capabilities and to (ii) further capability definitions composing the capability definitions. As a result, each capability definition provides the information regarding required resource types directly or indirectly for accomplishing goals of business processes. For example, when a required capability definition of a senior mobile software developer capability directly refers to five people possessing this capability and working in the respective organization, the capability definition will directly indicate the information regarding the possible resources for reaching the goals of the considered business process. Furthermore, a required capability definition of a project management capability without direct associations with any organizational resources will still require resources providing the capability. Thus, such a capability definition will indirectly provide the information regarding the required resources to business experts during the creation of resource-driven process definitions.

During $M_{2.1}$, business experts identify resource types of needed resources for accomplishing goals of the resource-driven process modeled with the help of the information contained in goal definitions. More specifically, they analyze capability definitions of goal definitions specified for resource-driven process definitions to identify resource types of the required resources. Capability definitions provide resourceful information for designing resource models of resource-driven process definitions. However, this information may not cover all resources actually needed to accomplish goals of business processes, as both goal definitions and capability definitions can be out-of-date. If business experts detect such missing resource types, they can update the corresponding goal definitions and capability definitions. For example, business experts can spot a resource type representing a new technology providing a required capability and associate the resource type with the definition of the respective capability. Similarly, when business experts recognize that accomplishing a goal aiming at a preliminary analysis of a mobile defect requires a senior developer capability, they can update the corresponding goal definition accordingly.

Based on the resource types identified, business experts add resource definitions to resource models of resource-driven process definitions during $M_{2.2}$. As explained in Section 3.3.3, resource types provide a reusable basis with customization points for configuring resource definitions specific to each resource-driven process. Specifically, resource types refer to a set of property definitions for describing customizable properties of the respective resources to be allocated for resource-driven processes, such as a property definition specifying customizable credentials of a Wiki service. During $M_{2.3}$, business experts set these properties of resource definitions added during $M_{2.2}$. For example, business experts customize a resource definition, representing a Wiki service using an admin username and password for accessing the service upon allocating it.

In addition, depending on the necessity of resources for resource-driven processes, *allocationBehavior* of resource definitions can be set as *uncoupled*, *coupled*, and *onDemand* during $M_{2.3}$. Required resource definitions must be allocated upon the initialization of business processes. Otherwise, the initialization of the corresponding resource-driven processes is terminated. For example, initializing a business process for fixing a mobile device defect without software developers will not make any sense. Therefore, business experts set the value of *allocationBehavior* as *coupled* for such important resource definitions. Alternatively, business experts can set the value of *allocationBehavior* as *uncoupled*. In this case, resources will be allocated upon initializing a resource-driven process definition similar to the case of *coupled*. In contrast to *coupled*, the initialization of resource-driven processes continues even if the allocation of resources specified as *uncoupled* fail. For example, if the allocation of a MediaWiki service fails, it can be configured later by allocated human resources. Thus, it will be wiser to proceed with the initialization of a resource-driven process containing the MediaWiki service, even if the allocation of the MediaWiki service somehow fails. Finally, business experts can set the value of *allocationBehavior* as *onDemand* to specify that the resource will be allocated after initialization in an ad hoc fashion. This allocation behavior enables the allocation of resources only if they are required during the execution of resource-driven processes. For example, a rare resource, such as 3-D printer, of an organization should be allocated on-demand (*onDemand*).

As described in Section 3.2, relationships among resources specify historical facts and desired facts about the resources they are connecting. For example, a “has-worked-together” relationship between two human resources specifies an additional criterion during the allocation of human resources. In contrast, a “managing” relationship between a human resource and IT resource will result in the execution of an additional business logic for adding a new user to the IT resource after allocating both of these resources at runtime. Business experts connect the resource definition with the relationship definition to detail resource models of resource-driven processes. To connect these, they first identify relationship types from the set of available types using their resource-driven process modeling environments during $M_{2.4}$. During the identification, they consider if the corresponding relationship fits the context of the resource-driven process definition being created/updated. For example, a “managing” relationship between a human resource and a project management service is needed only when certain resources should have restricted access to the project management service. Obviously, if every resource is allowed to access the project management service without any restrictions, it is not necessary to specify such a relationship. Moreover, during the selection of relationship types, business experts pay attention to target and source resource types of relationship types. For example, if business experts identify a “has-worked-together” relationship, they ensure that this relationship supports resource types included in the resource model.

After identifying appropriate relationship types, business experts connect resource definitions using relationship definitions created based on these relationship types during $M_{2.5}$. Similar to resource definitions, business experts configure additional properties of relationship definitions to customize relationships specified during $M_{2.6}$. Establishing each relationship can be either required or optional for considering a resource-driven process initialized. For example, if a project manager cannot access an allocated service that is highly relevant for managerial activities, it does not make sense to further execute a resource-driven process definition. Thus, business experts specify the corresponding use relationship between resource types representing the manager and the allocated service as required. In addition, in properties of this use relationship, business experts can specify access rights granted to the manager. After adding all relationships, the resource model is complete, and the resource-driven process definition is ready for execution. Furthermore, each goal definition, resource-driven process definition, resource definition, and relationship definition contain a set of instance descriptors (Definition 16). These instance descriptors can be used to specify due dates of the corresponding instances and their importance in the respective executions. For example, actors can mark an information resource as very important by setting its *importance* to 10. Similarly, actors can prioritize goals added

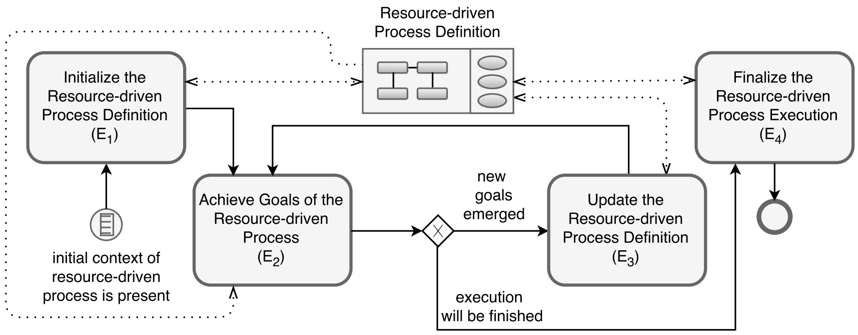

Resource-driven Process Definition
Initialize the Resource-driven Process Definition (E_1)
initial context of resource-driven process is present
Achieve Goals of the Resource-driven Process (E_2)
new goals emerged
Update the Resource-driven Process Definition (E_3)
execution will be finished
Finalize the Resource-driven Process Execution (E_4)

Figure 4.12: Steps in executing resource-driven processes.

using *importance* during the execution. Consequently, actors can make use of this information during future executions of the same resource-driven process. Next, we detail the execution of resource-driven processes.

4.3.3 Execute Resource-driven Processes (P_3) Using the Conceptual Framework

This phase of the Resource-driven Process Management (RPM) life cycle involves steps of executing resource-driven processes by initializing resource-driven process definitions created during P_2. Upon the presence of an initial context of a goal definition associated with a resource-driven processes definition, business experts can initialize the resource-driven process definition to achieve the goal, as illustrated in Figure 4.12.

Therefore, during the initialization of resource-driven processes (E_1), resource-driven process execution environments (Definition 42) execute the initialization algorithm presented in Algorithm 3. Moreover, the initialization algorithm of resource-driven process definitions has one input parameter containing a resource-driven process definition (p) to be initialized. The algorithm starts the initialization by creating a new instance descriptor representing the new resource-driven process at line 2. The created instance descriptor is added into the set of instance descriptors of the resource-driven

Algorithm 3 Initializing a resource-driven process definition.

```
1:  procedure initializeResourceDrivenProcess($p$)
2:      $i_p \leftarrow$ create a new instance descriptor for the resource-driven process with the state INITIALIZING
3:      $\mathcal{I}_{ad}[ad[p]] \leftarrow \mathcal{I}_{ad}[ad[p]] \cup \{i_p\}$
4:      for all $sdrd \in SD[rm[p]] \cup RD[rm[p]]$ do ▷ Create instance descriptors for resources and relationships
5:          $i_{sdrd} \leftarrow$ create a new instance descriptor with the state $PENDING$ $\wedge$ parent instance id $id[ie[i_p]]$ $\wedge$ source model id $id[ie[ad[sdrd]]]$
6:          $\mathcal{I}_{ad}[ad[sdrd]] \leftarrow \mathcal{I}_{ad}[ad[sdrd]] \cup \{i_{sdrd}\}$
7:      end for
8:      for all $id_g \in ID_g[p]$ do ▷ Create instance descriptors for goals
9:          $i_g \leftarrow$ create a new instance descriptor with the state $IN\_PROGRESS$ $\wedge$ parent instance id $id[ie[i_p]]$ $\wedge$ source model id $id_g$
10:         $\mathcal{I}_{ad}[ad[œ_{ME}(id_g)]] \leftarrow \mathcal{I}_{ad}[ad[œ_{ME}(id_g)]] \cup \{i_g\}$
11:     end for
12:     for all $sd \in SD[rm[p]]$ do ▷ Allocate interrelated resources
13:         if $allocationBehavior[sd]$ is not $onDemand$ then
14:             $allocateResource(sd, i_p, p)$
15:             if $state[i_p]$ is FAILED then
16:                 $finalizeResourceDrivenProcess(p, id[ie[i_p]], FAILED)$
17:                 break
18:             end if
19:         end if
20:     end for
21:     $state[i_p] \leftarrow INITIALIZED$
22: end procedure
```

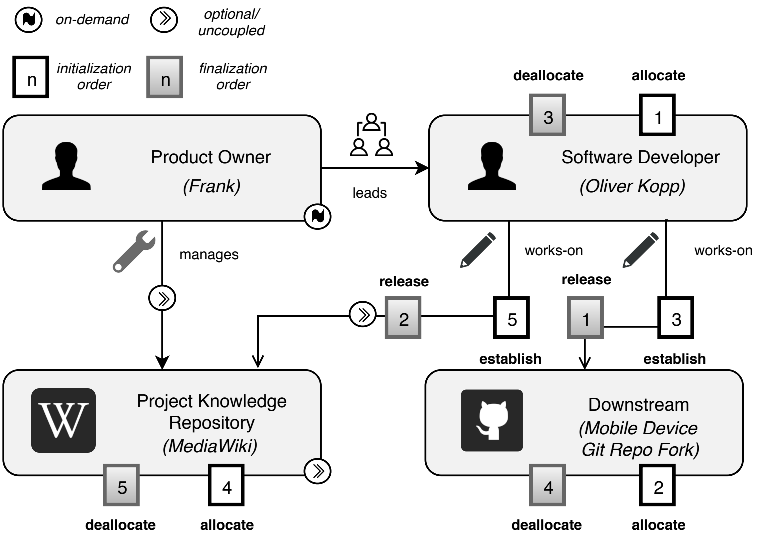


Figure 4.13: Order of allocation of resources and establishment of their relationships in a resource-driven process.

process definition at line 3. The creation and addition of the new instance descriptor are followed by the creation of an instance descriptor for each resource and relationship definition available in the resource model of the resource-driven process definition being initialized at line 5. To this end, each instance descriptor of a resource or relationship definition is created with (i) the state $PENDING$, (ii) the parent instance id referring to the instance descriptor (i_p) created for the resource-driven process, and (iii) the source model id indicating the resource or relationship definition ($sdrd$). Similarly, a new instance descriptor is created for each target goal of the resource-driven process at line 9 with (i) the state $IN\_PROGRESS$, (ii) the parent instance id referring to the instance descriptor (i_p) created for the resource-driven process, and (iii) the source model id indicating the target goal definition (id_g). To add the created instance descriptor, each goal definition is resolved from its entity identity using the entity identity resolver function (Definition 32) at line 10.

Thereafter, Algorithm 3 iterates through resource definitions available in the resource model of the resource-driven process definition at line 12 to allocate resource definitions without an *onDemand* allocation behavior, as specified at line 13. For example, the product owner in Figure 4.13 is not allocated because of his or her *onDemand* allocation behavior. Because of our assumption of passing variables using their references, changes made to the state of the resource-driven process definition after *allocateResource* will be visible in *initializeResourceDrivenProcess*. Thus, the progress of the initialization is checked after each call to *allocateResource* by controlling the state of the resource-driven process definition. If the resulting state of the resource-driven process definition is failed, the algorithm finalizes the execution using Algorithm 7 at line 16. Otherwise, the instance descriptor of the process is set to an initialized state at line 21.

Resource-driven process execution environments (Definition 42) allocate each resource definition of a resource-driven process definition as described in Algorithm 4. Furthermore, Figure 4.13 presents an example of such an allocation. Resource-driven process execution environments call *allocateResource* with three input parameters: (i) the resource definition (*sd*) to be allocated, (ii) an instance descriptor representing a resource-driven process (i_p), and (iii) the respective resource-driven process definition (p). The allocation starts with retrieving the corresponding instance descriptor (i_{sd}) specifying the instance of the resource definition in the resource-driven process using the function $\sigma_{\mathcal{I}}$. To call this function, entity identities referring to (i) the resource definition and (ii) the instance descriptor of the resource-driven process definition are given at line 2. After retrieving the corresponding instance descriptor, the retrieval of the appropriate operation endpoint capable of allocating resources automatically is done. Therefore, the algorithm creates a life cycle operation type (Definition 35) with the type ALLOCATE at line 3. The algorithm calls the operation endpoint selector function (Definition 39) with an entity identity referring to the resource type of the input resource definition (*sd*) and the life cycle operation type at line 4. The function σ_{OE} returns the appropriate operation endpoint ($oe_{lifecycle}$) capable

Algorithm 4 Allocate a resource definition.

```
1: procedure allocateResource($(sd, i_p, p)$)
2:     $i_{sd} \leftarrow \sigma_{\mathcal{I}}(id[ie[ad[sd]]], id[ie[i_p]])$ ▷ Get the corresponding instance descriptor of the resource definition
3:     $id_{lifecycle} \leftarrow$ a life cycle operation type ALLOCATE
4:     $oe_{lifecycle} \leftarrow \sigma_{OE}(id_{sType}[sd], id_{lifecycle})$ ▷ Select allocate life cycle operation endpoint of the corresponding resource type
5:     if $oe_{lifecycle}$ is not $\epsilon$ then ▷ Allocate resource, if possible
6:         $executeLifeCycleOperation(oe_{lifecycle}, id[ie[i_p]], sd, p)$
7:     else ▷ No such operation endpoint is available
8:         $state[i_{sd}] \leftarrow FAILED$ ▷ Set the state of the corresponding instance descriptor of the resource definition
9:     end if
10:    if $state[i_{sd}]$ is $ALLOCATED$ then
11:        $establishRelationships(sd, i_p, p)$ ▷ Establish relationships
12:    else $allocationBehavior[sd]$ is $coupled \wedge state[i_{sd}]$ is $FAILED$
13:        $state[i_p] \leftarrow FAILED$ ▷ Propagate failure
14:    end if
15: end procedure
```

of automatically allocating the resource out of the operation endpoints associated during the registrations of domain-specific information providers (Algorithm 1) and domain-specific operation providers (Algorithm 2), if such an operation endpoint is available. Otherwise, the function returns the value ϵ and the allocation fails at line 8. For example, the function returns an allocation operation endpoint representing a web service capable of communicating with a software developer and allocating the developer for the resource-driven process.

If an operation endpoint was found, the allocation operation is executed using the life cycle operation endpoint at line 6. For example, calling such an allocation operation endpoint of a Wiki service can result in deploying the CSAR of the Wiki service on an OpenTOSCA container. As shown at line 6, the function for calling a life cycle operation endpoint takes four input parameters: (i) the life cycle operation endpoint to be called, (ii) the entity identity of the instance descriptor representing the resource-driven process containing the resource definition being allocated, (iii) the resource or relationship definition, and (iv) a resource-driven process definition. Thus, the web service referred to by the operation endpoint can use all the information available in the current resource model during the allocation, such as relationships of the allocated resource. Moreover, the corresponding resource-driven process execution environment should be capable of executing the operation endpoint. For example, all required operation types (ID_{oe}) of an operation endpoint should be available on the respective resource type. Furthermore, non-functional properties for calling life cycle operations, such as the number of allocation trials made for a MediaWiki resource before setting the state to $FAILED$, are specified in respective operation endpoints. The details of calling operation endpoints are beyond the scope of this work and therefore will not be discussed further. The execution of the allocation operation updates the instance descriptor (i_{sd}) representing the instance of the allocated resource at line 6. For example, it can change the state of the instance descriptor to the allocated state, if the operation was successful. If the allocation of a resource has failed, and the resource is set as a required resource by setting the allocation behavior to *coupled*, the state of the instance descriptor describing the parent resource-driven process containing the definition of the resource is set to failed at line 13.

If the allocation of a resource is successful, the algorithm proceeds with establishing relationships connected to the resource in the resource model of the resource-driven process definition using Algorithm 5. Therefore, Algorithm 5 retrieves all relationship definitions connected to the resource definition (sd) using the relationship retriever function (Definition 40). For example, after allocating software developer in Figure 4.13, the two “works-on” relationships are retrieved. Thus, *establishRelationships* iterates through all relationship definitions at line 3. For each relationship definition, the algorithm first retrieves its instance descriptor (i_{rd}) created in the context of an instance (i_p) of a resource-driven process using $\sigma_{\mathcal{I}}$, as shown at line 4 of

Algorithm 5 Establish relationship definitions.

```
procedure $establishRelationships(sd, i_p, p)$
    $id_p \leftarrow id[ie[i_p]]$ ▷ Get entity identity of the process instance
    for all $rd \in \sigma_{RD}(sd, rm[p])$ do ▷ Establish every pending incoming and outgoing relationship from given resource if both resources connected by the relationship are allocated
        $i_{rd} \leftarrow \sigma_{\mathcal{I}}(id[ie[ad[rd]]], id[ie[i_p]])$
        if $state[i_{rd}]$ is $PENDING \wedge state[\sigma_{\mathcal{I}}(id_{sdSource}[rd], id_p)]$ is $ALLOCATED \wedge state[\sigma_{\mathcal{I}}(id_{sdTarget}[rd], id_p)]$ is $ALLOCATED$ then
            $id_{lifecycle} \leftarrow$ a life cycle operation type ESTABLISH
            $oe_{lifecycle} \leftarrow \sigma_{OE}(id_{rType}[rd], id_{lifecycle})$
            if $oe_{lifecycle}$ is not $\epsilon$ then ▷ Operation endpoint exists?
                executeLifeCycleOperation$(oe_{lifecycle}, id_p, rd, p)$
            else ▷ No such operation endpoint is available
                $state[i_{rd}] \leftarrow FAILED$
            end if
            if $mandatory[rd]$ is true and $state[i_{rd}]$ is $FAILED$ then
                $state[i_p] \leftarrow FAILED$ ▷ Propagate failure
                break
            end if
        end if
    end for
end procedure
```

Algorithm 5. As a relationship can only be established between two allocated resources once, the algorithm first determines (i) if the relationship is not yet established and (ii) if resources connected by the relationship are already allocated at line 5. For example, after allocating a software developer in Figure 4.13 the relationships “works-on” cannot be established, as the project knowledge repository and downstream Git repository are not yet allocated.

For each relationship definition (*rd*), the operation endpoint for automatically establishing the respective relationship is selected at line 7. If the life cycle operation endpoint is available, it is called for the automated establishment of the relationship at line 9. If the operation endpoint for establishing the relationship (i) finishes unsuccessfully or (ii) is unavailable, the state of the instance descriptor (i_{rr}) of the relationship definition fails. Furthermore, if the establishment of a required relationship fails, the algorithm sets the state of instance indicating the resource-driven process to failed at line 14. As a result of executing the algorithm shown in Algorithm 5, all pending relationships between a given and previously allocated resource definition are established. After the successful allocation of every resource in a resource model, a resource-driven process is considered initialized.

If the initialization succeeds, phase P_3 continues with step E_2, as shown in Figure 4.12. During this step, allocated actors of a resource-driven process work toward the goals of the resource-driven process using other allocated resources. Therefore, actors first prioritize goals of the resource-driven process using the *priority* field of the created instance descriptors for goal definitions in a resource-driven process modeling tool (Definition 31). Moreover, actors assign due dates for the target goals of the resource-driven process. As certain resources will be needed for particular target goals, actors can similarly set due dates of instance descriptors of the resources. For example, a target goal aimed at analyzing impact of a mobile device defect within the three days can require a simulation machine capable of analyzing different components. After expiration of this due date, the simulation machine will not be required anymore. Therefore, actors set due dates of the target goal and the simulation machine accordingly. Moreover, actors mark important resources or relationships during the execution using the *priority* field of the instance descriptors of resources and relationships, so that they can be identified easily in future executions. For example, an allocated software developer can be marked important if he has fixed a critical defect during a business process aimed at fixing mobile device defects. Goals of resource-driven processes change during their execution. Consequently, business experts update resource-driven process definitions based on the changing goals as detailed in Section 4.3.2. For example, business experts add new goal definitions or new resource definitions to resource-driven process definitions.

As a result of updating resource-driven process definitions, resources may be allocated or deallocated. After accomplishing all goals specified in resource-driven process definitions, business experts initiate the finalization of resource-driven process executions during E_4. Moreover, all relationships of allocated resources are released, and the resources are deallocated. Algorithm 6 presents the detailed business logic of the deallocation of a resource and the release of its relationships. The algorithm is called with three input parameters: (i) the resource definition to be deallocated (sd), (ii) the instance descriptor representing a resource-driven process (i_p), and (iii) the respective resource-driven process definition containing the resource definition and instance descriptor (p). The deallocation aims at reversing the effects of the allocation algorithm presented previously in Algorithm 4. Therefore, the first effects of relationships are reversed by calling the operation endpoints specific for releasing the relationships. For example, before deallocating a software developer his or her access rights are removed from a project knowledge repository and a downstream Git repository, as illustrated in Figure 4.13. Naturally, each release operation can be called only once between two allocated resources. Therefore, the *deallocateResource* algorithm checks (i) if both resources connected by a relationship are still allocated and (ii) if the relationship is still established at line 7. To release relationships fulfilling this precondition, Algorithm 6 first selects a release operation endpoint ($oe_{lifecycle}$) for each relationship definition at line 8 using the life cycle operation type ($id_{lifecycle}$) created at line 3. The life cycle operation endpoint is called for releasing the relationship at line 10, if available. Please note that releasing an established relationship is optional; thus, the unavailability of an operation endpoint for release does not lead to a failed state. When an established relationship between an allocated resource and a resource with an unknown state exists (Algorithm 6), the effects of the relationship will possibly remain, such as keeping access rights of a manager on a resource. To avoid problems in such cases, the technical experts are informed.

After releasing all relationships, the deallocating operation endpoint is selected at line 19. The life cycle operation endpoint is called for deallocating the corresponding instance of the resource 21. If a deallocation operation

Algorithm 6 Deallocate a resource definition.

```
1: procedure $deallocateResource(sd, i_p, p)$
2:     $id_p \leftarrow id[ie[i_p]]$ ▷ Get entity identity of the process instance
3:     $id_{lifecycle} \leftarrow$ a life cycle operation type RELEASE
4:     for all $rd \in \sigma_{RD}(sd, rm[p])$ do ▷ Release all established relationships
5:         if $state[\sigma_{\mathcal{I}}(id[ie[ad[rd]]], id_p)]$ is $ESTABLISHED \wedge$
6:           $state[\sigma_{\mathcal{I}}(id_{sdSource}[rd], id_p)]$ is $ALLOCATED \wedge$
7:           $state[\sigma_{\mathcal{I}}(id_{sdTarget}[rd], id_p)]$ is $ALLOCATED$ then ▷ Release only established relationships between two allocated resources
8:             $oe_{lifecycle} \leftarrow \sigma_{OE}(id_{rType}[rd], id_{lifecycle})$
9:             if $oe_{lifecycle}$ is not $\epsilon$ then ▷ Operation endpoint exists?
10:                $executeLifeCycleOperation(oe_{lifecycle}, id_p, rd, p)$
11:            end if
12:        else if $state[\sigma_{\mathcal{I}}(id[ie[ad[rd]]], id_p)]$ is $ESTABLISHED \wedge$
13:                $(state[\sigma_{\mathcal{I}}(id_{sdSource}[rd], id_p)]$ is $UNKOWN \vee$
14:                $state[\sigma_{\mathcal{I}}(id_{sdTarget}[rd], id_p)]$ is $UNKOWN)$ then ▷ The relationship is established but one resource has an unknown state
15:            $informATechnicalExpert(id_p, rd, UNKOWN)$ ▷ Inform a technical expert about this unknown state
16:        end if
17:    end for
18:    $id_{lifecycle} \leftarrow$ a life cycle operation type DEALLOCATE
19:    $oe_{lifecycle} \leftarrow \sigma_{OE}(id_{sType}[sd], id_{lifecycle})$
20:    if $oe_{lifecycle}$ is not $\epsilon$ then ▷ Operation endpoint exists?
21:        $executeLifeCycleOperation(oe_{lifecycle}, id_p, sd, p)$
22:    else ▷ No such operation endpoint is available
23:        $state[\sigma_{\mathcal{I}}(id[ie[ad[sd]]], id_p)] \leftarrow FAILED$ ▷ Set the corresponding instance descriptor of the resource definition
24:        $informATechnicalExpert(id_p, sd, MISSING\_OP)$ ▷ Inform a technical expert about the missing operation
25:    end if
26: end procedure
```

for the resource type is not available, the state of the instance descriptor retrieved is set to failed at line 23. After setting the state to failed, a technical expert is informed. Consequently, each allocated resource is first cleaned from the effects of a relationship and then deallocated. For example, before deallocating the software developer in Figure 4.13, the "works-on" relationships are released. Notably, the deallocation of information resources typically allows the information to be stored with less resources consumed, so that the information can be reused in a future execution.

Algorithm 7 Finalize a resource-driven process.

```
1:  procedure $finalizeResourceDrivenProcess(p, i_p, state_{new})$
2:      for all $sd \in SD[rm[p]]$ do
3:          if $state[\sigma_{\mathcal{I}}(id[ie[ad[sd]]], id[ie[i_p]])]$ is $ALLOCATED$ then ▷ Deallocate only resources that are still allocated
4:              $deallocateResource(sd, i_p, p)$
5:          end if
6:      end for
7:      for all $id_g \in ID_g[p]$ do ▷ Update states of target goal instances
8:          $state[\sigma_{\mathcal{I}}(id_g, id[ie[i_p]])] \leftarrow state_{new}$
9:      end for
10:     $state[i_p] \leftarrow state_{new}$ ▷ Update the state of the process instance
11: end procedure
```

The algorithm for finalizing resource-driven process executions makes use of this deallocation algorithm, as shown in Algorithm 7. The finalization algorithm of resource-driven process definitions is called with (i) a resource-driven process definition (p) containing one or more instance descriptors, (ii) the instance descriptor of the resource-driven process definition to be finalized (i_p), and (iii) the final state of the instance descriptor ($state_{new}$). More specifically, it iterates through all resource definitions included in the resource model of the resource-driven process (p) being finalized at line 2. Furthermore, the algorithm deallocates every resource in an allocated state using the deallocation algorithm at line 4. After iterating through all resource

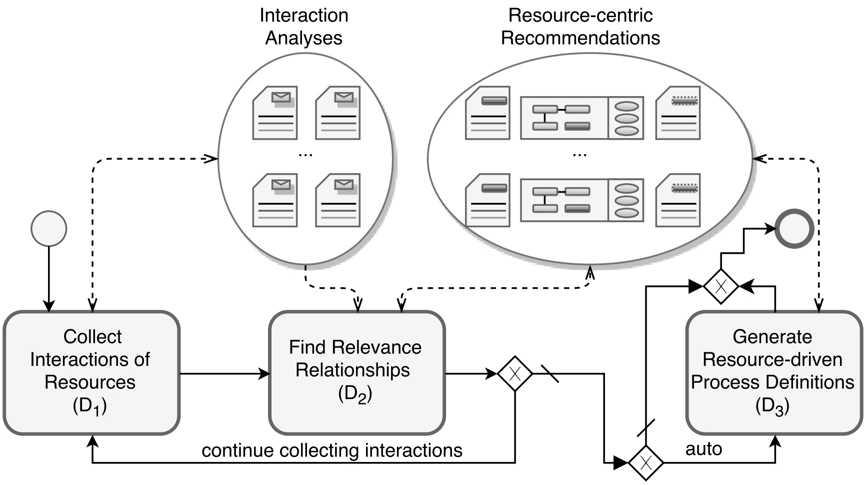

Interaction Analyses
Resource-centric Recommendations
...
...
Collect Interactions of Resources (D$_1$)
Find Relevance Relationships (D$_2$)
Generate Resource-driven Process Definitions (D$_3$)
continue collecting interactions
auto

Figure 4.14: Steps in generating resource-driven process recommendations.

definitions, the instance descriptors of goal definitions and the resource-driven process (i_p) is updated with the given state at lines 8 and 10. This results in the conclusion of a resource-driven process execution, shown in Figure 4.12 in E_4. The main complexity of the presented algorithms resides behind the calls for allocating / deallocating resources and establishing / releasing relationships. These calls block the execution of the presented algorithms and take time to execute. Next, we present the fourth phase of the RPM life cycle.

4.3.4 Discover Resource-driven Process Enactments (P_4) Using the Conceptual Framework

Executing resource-driven processes leaves different types of interaction logs, such as emails in mailing lists, Git interactions, and Wiki entries. Collecting and interpreting these interactions can help organizations identify the actual resources and their capabilities used during executions of resource-driven processes. Using these identified resources and capabilities, business experts can improve existing resource-driven process definitions at modeling time (P_2). Therefore, resource-driven process discovery environments (Definition 54) execute the steps shown in Figure 4.14 automatically to generate resource-centric process recommendations including relevance relationships (Definition 48) and new resource-driven process definitions based on the relevance relationships.

Business experts can repeatedly initialize a resource-driven process definition to reach the addressed goals in the resource-driven process definition. Consequently, each instance of a resource-driven process will be described using an instance descriptor. Moreover, instance descriptors stored under resource definitions represent different allocated resources for accomplishing goals of business processes. Interestingly, actual executions of resource-driven processes can diverge from resource-driven process definitions in terms of their allocated resources and capabilities. For example, an external expert can fix a software defect in a mobile device in an ad hoc fashion during a business process aimed at fixing a mobile device defect. Furthermore, documenting this external expert in the corresponding resource-driven process definition can improve the performance of future executions, in case a defect is spotted in the software parts changed by the external expert. Consequently, recommending the inclusion of types of resources positively affecting resource-driven processes to business experts P_2 can both (i) ease their decision-making process and (ii) result in the creation of better-performing resource-driven processes in terms of the time and quality of outcomes. Similarly, recommending the exclusion of types of resources that were not used at all can result in resource-driven processes with decreased resource consumption and, thereby, increased resource utilization. To identify types of resources that can be included in resource-driven processes or excluded from them, interaction collectors (Definition 43) gather interaction analyses (Definition 44) of resources allocated using available interaction logs. During the gathering of interactions, interaction collectors make use of all instance descriptors of resource definitions in resource models of resource-driven processes. Thus, they aggregate all available instance descriptors available under resource definitions in resource models of resource-driven processes to collect their interactions from different interaction logs.

After creating a collection containing all instance descriptors, resource-driven process discovery environments use the interaction analysis generator function (Definition 46) and registered interaction collectors during P_1 for deriving interaction analyses of each allocated resource (D_1). Each interaction analysis contains an instance descriptor representing a relevant resource (Definition 45). Using instance descriptors of relevant resources, resource-driven process discovery environments can gather further interaction analyses, as shown in Figure 4.14. For example, they can analyze interactions of the software developer committed to the Git repository. Consequently, during D_1 of P_4, resource-driven process discovery environments use the interaction analysis generator function (Definition 46) iteratively for gathering interaction analyses of allocated resources and identified relevant resources using interaction collectors.

Interestingly, resource-driven process discovery environments can use interaction analyses to identify relevant resources (Definition 45) and relevant capabilities (Definition 47). Moreover, each relevant resource and capability is related to a business process differently. For example, including a software developer with a high number of contributions can influence the execution of a resource-driven process positively (a positive relevance). In contrast, including a user constantly sending spam emails in a future resource-driven process execution will probably result in a negative influence on the execution of the respective resource-driven process (a negative relevance). To describe different types of relevance, we introduce the concepts of relevance relationships (Definition 48). Relevance relationships have correlation coefficients describing the type of relevance between relevant resources or relevant capabilities and resource-driven processes. To derive relevance relationships of interaction analyses collected during D_1, resource-driven process discovery environments use the relevance relationship generator function (Definition 50) with available relevance mappers (Definition 49). For example, the function generates a relevance relationship containing a software developer associated with a positive correlation coefficient using an interaction analysis of a Git commit interaction. Moreover, if a relevant resource contains certain capabilities, these capabilities can be relevant, too. Generated relevance relationships contain such relevant capabilities with their correlation coefficients. For example, the function will generate a relevance relationship containing a relevant software development capability associated with a positive correlation coefficient using an interaction analysis of a commit interaction collected by analyzing a Git repository. The resulting relevance relationships provide recommendations containing positive, negative, or no correlated relevant resources and relevant capabilities for business experts to model resource-driven processes (P_2).

Using derived relevance relationships and the resource-driven process definition used to generate relevance relationships, it is further possible to create new resource-driven process definitions containing these relevance relationships. For example, it is possible to create a resource-driven process definition containing all relevant resources and capabilities with a correlation coefficient higher than 0.8. Such an automated generation of resource-driven process definitions can reduce the modeling effort of business experts. To create resource-driven process definitions based on generated relevance relationships and resource-driven process definitions used to generate these, resource-driven process discovery environments use the resource-driven process definition generator function (Definition 52) with a set of resource-driven process definition recommenders (Definition 51) during D_3. As shown in Figure 4.14, this step is optional. After generating these new resource-driven process definitions containing relevant resources and relevant capabilities, business experts control the generated resource-driven process definitions and update them if needed. Consequently, the automated generation of resource-driven process definitions does not remove the phase P_2 but only aims at reducing the effort of business experts. After this phase, P_1 is executed again to adapt modeling, execution, and discovery environments of resource-driven processes to the changing organizational conditions. Next, we present a discussion of the RPM life cycle detailed in this chapter.

4.4 Discussion

The RPM life cycle presents four phases for using resource-driven processes in organizations. In general, the presented life cycle resembles the main characteristics of the BPM life cycle presented in Section 2.2. A major difference between the two life cycles is the ordering of the preparation and configuration phases involving activities for creating IT services for executing business processes. The reason for this difference is that the identification of resources and their relationships provides sufficient information for the preparation phase for resource-driven processes. However, in activity-oriented business processes, a more detailed modeling is required for starting with the preparation phase (the configuration phase in Section 2.2). Another reason for this reordering of the phases is the nature of resource-driven processes targeted by them. Informal processes can be emergent and cannot be foreseeable. Consequently, the IT services capable of supporting and automating them should already be available before modeling them, P_1. As a result, business experts can create definitions of resource-driven processes for reactively handling unanticipated situations and can directly initialize them. Furthermore, the configuration phase of the BPM life cycle focuses on the development of necessary IT infrastructure capable of automating interrelated activities modeled previously. The preparation phase of the RPM life cycle aims at establishing an IT infrastructure capable of (i) automating the allocation and deallocation of interrelated resources and (ii) analyzing enactments of resource-driven processes. Consequently, during the enactment of resource-driven processes, interrelated resources are automatically allocated, similarly to the service templates modeled in TOSCA. For example, the presented initialization algorithm (Algorithm 3) resembles the algorithm shown by Breitenbücher et al. [BBK+14; BBK+16] for automated generation of business processes capable of initializing services. The presented initialization algorithm of resource-driven processes (Algorithm 3) directly invokes the allocation operations, whereas the algorithm presented by Breitenbücher et al. [BBK+14; BBK+16] focuses on generating a business process model capable of initializing a service. Furthermore, our algorithm does not focus on the types of relationships but on the types of operation endpoints.

The P_4 of the RPM life cycle can be considered the evaluation phase of the BPM life cycle involving process mining activities. Notably, during the generation of resource-driven recommendations, we do not assume process-aware execution systems. Consequently, the event logs used during the generation of resource-centric recommendations do not refer to business process instances or activities. In addition, the generated resource-centric recommendations are different from expert recommendations due to recommending not only human but also other resources, such as the expert recommendation approaches presented by Dorn et al. [DD11], Schall [Sch12], and Wang et al. [WZN16] explained in Section 2.2. Furthermore, presenting correlation coefficients of resources to resource-driven processes distinguishes our approach for generating recommendations from, for example, the Codebook approach [BKZ10] to analyzing interactions of software development repositories.

Chapter 5

Incorporating Resource-driven Processes into Activity-oriented Business Processes

Organizations can model and execute their business processes using resource-driven processes. Furthermore, the resource-driven modeling and execution of business processes give organizations the opportunity to execute business processes without relying on their predefined activities. Thus, resource-driven modeling and execution of business processes are suitable for business processes with unpredictable activities at modeling time (informal processes; Section 1.1). Moreover, resource-driven processes present a generic concept of resources including IT resources, such as web services implementing business processes modeled and executed in activity-oriented fashion. Thus, business experts can add activity-oriented processes into resource-driven process definitions. For example, a simulation process defined in Business Process Model and Notation (BPMN) [Obj11] or WS-Business Process Execution Language (BPEL) [OAS07b] can be included in a resource-driven process definition along with an electronic engineer responsible for evaluating the results of the simulation process. Thus, it is possible to incorporate business processes modeled in an activity-oriented fashion into resource-driven processes by implicitly representing these as resources. Upon the initialization of a resource-driven process, activity-oriented business processes will be automatically allocated and will be executed as prescribed in their business process models.

Furthermore, in various business processes, such as manufacturing processes, tasks are executed manually (Section 1.2.3). For example, a subprocess can be the execution of a maintenance process in a manufacturing process manually with the help of additional resources in case an error is detected. Because of the suitability of resource-driven processes for representing informal processes in a resource-driven way, business experts can express these manual parts of activity-oriented business processes using resource-driven process definitions. In this chapter, we concentrate on incorporating resource-driven processes in business processes modeled in an activity-oriented fashion, such as BPMN or BPEL processes (Contribution 5). Therefore, we first investigate a set of requirements for incorporating resource-driven processes into structured processes (Section 5.1). We introduce the concept of Context-sensitive Activity (CsA) meeting the requirements presented in Section 5.2 and we conclude with a discussion in Section 5.3.

5.1 Requirements for Incorporating Resource-driven Processes into Activity-oriented Business Processes

A straightforward method to incorporate resource-driven processes into business processes modeled in an activity-oriented fashion includes defining resource-driven processes as a web service and invoking these as activities in business processes. For example, if a machine fails, a business process execution engine makes a service call to initialize a resource-driven process aimed at repairing the failed machine. Interestingly, in various cases, the decision to execute a business process manually or automatically depends on the actual context (Section 2.3.1.1). For example, the investigation of a mobile device defect can be done automatically with a simulation machine running diagnostics to identify the cause of errors. If such a machine fails, the diagnostics should be conducted manually, so that the defect can be fixed. Consequently, an activity-oriented business process can select the automated business process alternative or the manual process based on the actual context of the organization. To consider such situations, instead of incorporating resource-driven processes straightforwardly by initializing them with service calls, we investigated different aspects of activity-oriented business processes, as explained in Section 2.3.1.1. Based on these aspects, we derived a set of requirements for incorporating resource-driven processes into business processes modeled and executed in an activity-oriented fashion. We do not claim that this set of requirements is complete, but it is a starting point for further research. Next, we describe each requirement.

5.1.1 Enabling Context-sensitive Variation of Activity-oriented Business Processes (Ri_1)

The contextual data of business processes characterize situations of business processes using the information of any relevant entity, such as the information regarding the current demand of a product manufactured in a production process. Moreover, the adoption of the Internet of Things technologies enables organizations to enrich the contextual data of business processes with information from real-world entities, such as the physical condition of simulation devices used in a business process. Using the available contextual data, business processes can adapt their business logic dynamically and can become context-sensitive. For example, a business process can initiate a resource-driven process for maintaining their simulation machines based on the predictive maintenance analysis generated using the data collected from sensors deployed on the simulation machines [Mob02; OAÇ06]. Therefore, business experts should be able to define the context in business process models of activity-oriented business processes. Moreover, the represented context should allow adaptations based on the current contextual data of the respective business process at runtime (derived from Section 2.3.1.1).

5.1.2 Enabling Goal-oriented Adaptation of Activity-oriented Business Processes (Ri_2)

Technological advancements can result in alternative methods for reaching the same goals. For example, an organization can create a semiautomated business process by creating an alternative method for automating the investigation part of a business process aimed at investigating and fixing a mobile device defect. Such semiautomated alternative business processes will start with an automated diagnostic process investigating the problem using simulation machines designated for this purpose. After simulations complete their diagnostics, a group of software developers and embedded system developers analyze the results to take the necessary actions. Moreover, the respective organization opts for this semiautomated alternative only if the simulation machines are functioning properly. Otherwise, the organization can fall back to the manual business process aimed at investigating and fixing a mobile device defect. Consequently, organizations can enact their business processes uninterruptedly, which plays a role in today's organizations [Rot16; Lan+13; EL16]. Furthermore, organizations can execute an additional business logic to ensure the uninterrupted enactment of other business processes. For example, to ensure uninterrupted execution of a semiautomated business process aimed at investigating and fixing a mobile device defect, organizations need to ensure that their simulation machines function properly. To do this, they can run predictive failure analyses using the data from the sensors deployed on these machines to maintain them before they break down. If the need for maintenance is detected, the organization can initiate a resource-driven process aimed at maintaining the machines in parallel at runtime to avoid any outages. Executing the additional business logic requires an automated selection of the business logic to reach organizational goals context-sensitively. Such an automated selection will use the available contextual data to select the right business logic, such as the decision between a semiautomated and manual business process based on the state of simulation machines. Moreover, such an automated selection should enable running the necessary preventive business logic in parallel to avoid any breakdowns (derived from Section 2.3.1.1).

5.1.3 Enabling Optimal Adaptation of Activity-oriented Business Processes (Ri_3)

Business experts create business process models to capture recurring interrelated activities of different executions of the same business process. Consequently, the business process models are typically modeled generically, addressing as many situations as possible. Thus, depending on the actual situation of an organization, it can be possible to adjust the business process model to be executed more optimally, if applicable. For example, a resource-driven process definition aimed at fixing a mobile device defect can contain five human resources to resolve the defect. Moreover, the corresponding organization enacting this resource-driven process can recognize that, for defects originating from less than 20 mobile returns, it is sufficient to execute a business process with two human resources. Organizations may detect such returned mobile devices using integrated RFIDs [HZJ08; HZJ08]. Another example of an optimization before executing a resource-driven process is configuring workplaces for allocated human resources of resource-driven processes automatically to provide them ergonomic working conditions [DWHB15; MR14]. To optimize business processes, such as avoiding exhaustive resource usage, there is a need to automatically optimize business processes based on the current contextual data. Consequently, organizations can automatically optimize their business processes, such as decreasing the number of human resources depending on the contextual data (derived from Section 2.3.1.1). Next, we present a new activity type for business processes meeting the requirements described here.

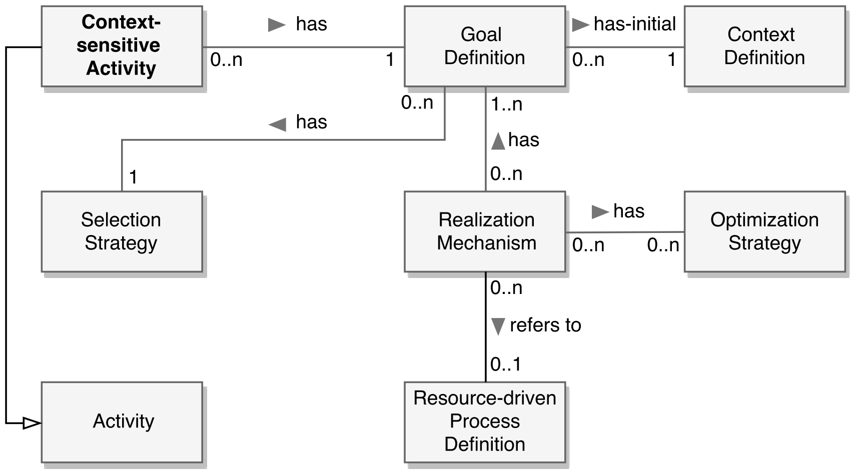


Figure 5.1: Meta-model of Context-sensitive Activities.

5.2 Context-sensitive Activities

To address the requirements presented in Section 5.1, we propose a new type of activity: a Context-sensitive Activity (CsA) capable of making decisions based on the available contextual data. Consequently, business experts can incorporate resource-driven process definitions in a context-sensitive fashion using CsA. Moreover, depending on the contextual data, CsA can opt for executing resource-driven processes in parallel with the main business logic achieving the goal of a CsA. For example, simulation machines used in a business process can be maintained based on the results of a predictive maintenance analysis in parallel using a resource-driven process to ensure continuous enactment of the business process. In Section 5.2.1, we describe the concept of this new type of activity. Hereafter, we present the execution semantics of the new activity in Section 5.2.2.

5.2.1 Meta-model of a Context-sensitive Activity

Resource-driven process definitions aim at specifying the implicit business logic by allocating resources capable of achieving the goals of a business process. To incorporate these resource-driven process definitions into activity-oriented business process models aligned with the requirements presented in Section 5.1, we introduce a new type of activity called CsA.

Definition 56 (Context-sensitive Activity (informal)). A Context-sensitive Activity (CsA) is a specific type of activity in an activity-oriented business process model enabling the context-sensitive selection of the business logic defined through realization mechanisms (Definition 57) aimed at different goals. ♣

We base the definition of CsA on the formal definitions of activities presented in Leymann and Roller [LR00]. Thus, the resulting CsA map between input containers and output containers during execution of a business process as explained by [LR00]. In addition, it uses additional information, such as contextual data, to generate output containers. More specifically, Figure 5.1 gives an overview of the meta-model of each Context-sensitive Activity.

Each CsA aims at the accomplishment of a goal specified with a goal definition. Goal definitions presented in Figure 5.1 extend goal definitions of Resource-driven Process Modeling Language (Redo) presented in Section 3.3.3. Thus, each goal definition describes its initial context and final context using context definitions presented in resource-driven process modeling language (Definition 27). Initial contexts of goal definitions describe situations in which organizations need to achieve the respective goals. For example, the execution of a business process aimed at investigating and fixing a mobile device defect will start upon the presence of a defect. Thus, referencing a goal definition in each CsA describes the situation in which the respective CsA needs to be activated. To achieve the goal of a CsA, business experts define realization mechanisms aimed at these goals.

Definition 57 (Realization Mechanism (informal)). Realization processes bind existing business process models to their goal definitions and provide sufficient information for creating instances of the represented business processes, such as the location of business process models, their types, and default initialization parameters. Moreover, realization mechanisms may refer to optimization strategies for associated business process models.♣

For example, a realization mechanism can refer to a resource-driven process definition aimed at investigating and fixing a mobile device defect. To this end, it must provide necessary information for executing the respective resource-driven process definition or any other referenced business process model. Please note that a discussion of related approaches is found in Section 5.3. For example, in a resource-driven process definition, it is sufficient to have a unified resource identifier of the respective resource-driven process definition. In contrast, in a BPEL process, a zip file containing a deployment descriptor will be needed for the initialization. Thus, a realization mechanism does not contain the actual activity implementation but, rather, it provides the necessary information required for executing an activity implementation in a unified form. The necessary information includes goal definitions targeted by a realization mechanism, as shown in Figure 5.1. Moreover, to select one realization mechanism out of multiple realization mechanisms indicating the same goal definition, each goal definition has a selection strategy.

Definition 58 (Selection Strategy (informal)). Selection strategies are automated software services taking multiple realization mechanisms and the available contextual data as input and returning one of the realization mechanisms as output. Selection strategies are a special case of service discovery. ♣

Consequently, a business process execution engine supporting CsA will use these automated selection strategies to choose between different types of realization mechanisms based on the available contextual data. For example, if two resource-driven process definitions aimed at investigating and fixing a mobile device defect exist and if functioning simulation devices capable of automatically executing the investigation of the resource-driven process exist, the selection strategy will select the semiautomated version. Consequently, technical experts specify selection strategies to ensure that one realization mechanism aimed at the specified goal for a CsA will be selected based on the available contextual data. The details of the selection will be explained in Section 5.2.2. Moreover, each realization mechanism can have multiple optimization strategies.

Definition 59 (Optimization Strategy (informal)). Optimization strategies are automated software services taking a realization mechanism and the contextual data available as input and applying certain optimization measures to generate an optimized version of a business process model associated with the realization mechanism. ♣

For example, after selecting a semiautomated business process, the optimization strategy can decrease the number of software developers based on the number of returned mobile devices. Consequently, technical experts specify optimization strategies to ensure that an optimal variant of a realization mechanism will be executed depending on the available contextual data. Please note that optimization strategies are dependent on existing optimization techniques and cannot do better than these. Next, we describe the execution semantics of a CsA.

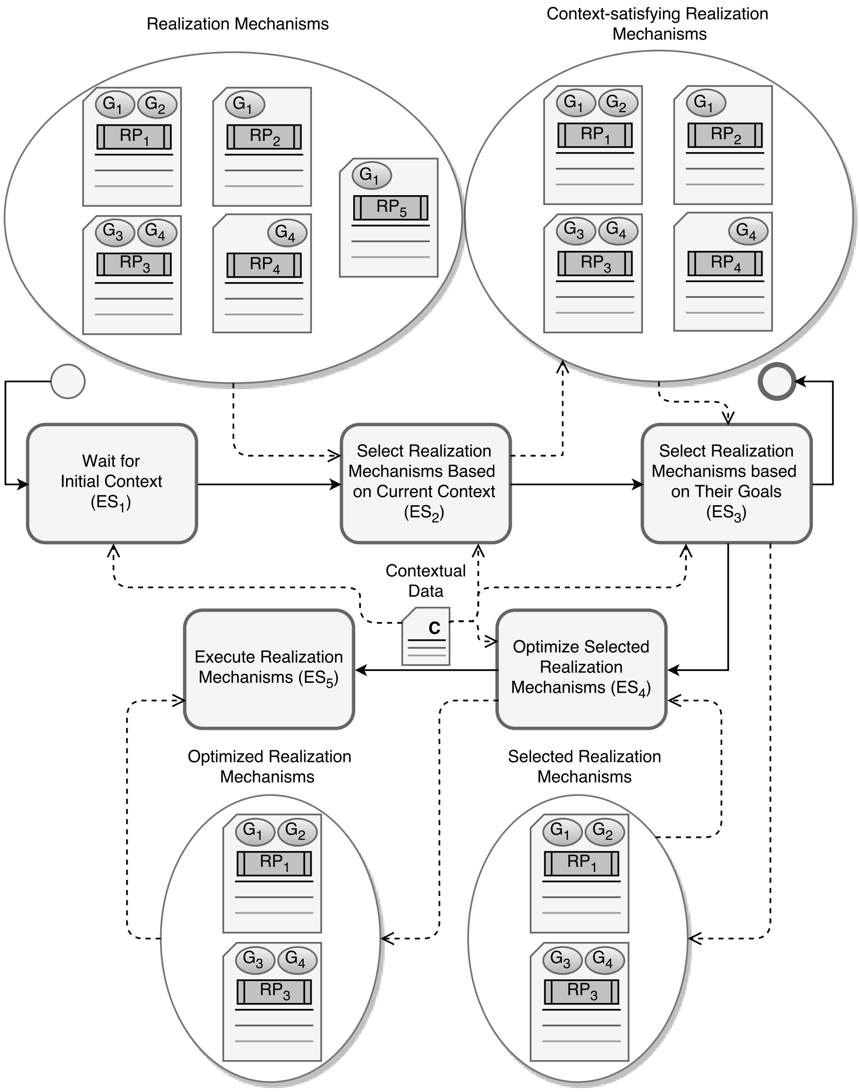

Realization Mechanisms
Context-satisfying Realization Mechanisms
G1 G2 RP1
G1 RP2
G1 RP5
G3 G4 RP3
G4 RP4
Wait for Initial Context (ES1)
Select Realization Mechanisms Based on Current Context (ES2)
Select Realization Mechanisms based on Their Goals (ES3)
Contextual Data
C
Execute Realization Mechanisms (ES5)
Optimize Selected Realization Mechanisms (ES4)
Optimized Realization Mechanisms
Selected Realization Mechanisms

Figure 5.2: Illustration of execution semantics of a Context-sensitive Activity aimed at fixing a mobile device defect (G_1).

5.2.2 Execution Semantics of a Context-sensitive Activity

The presented new type of activity, CsA, enables incorporating resource-driven process definitions based on their goal definitions containing their initial context definitions. To automate the recognition of certain contexts based on available contextual data, technical experts describe context definitions of CsA using machine-processable languages, such as an XPath expressions. For example, a context definition can describe the situation of returned mobile devices using the following XPath expression: $count(/\$contextualData/ReturnedMobileDevice) > 0$. Furthermore, business process execution engines supporting CsA have access to the current contextual data containing any information relevant for describing the situation of the executed business process (i.e., a context system, such as the Nexus platform [LCG+09], is part of the environment). For example, such contextual data can contain information regarding the status of simulation machines and available software developers. Moreover, a context definition can describe the initial state of a CsA aimed at fixing a mobile device defect by expressing the condition of returned mobile devices using the available variables in the contextual data.

Consequently, business process execution engines supporting CsA can evaluate context definitions using the available contextual data to understand whether the current context matches a context specified in a context definition, such as checking whether the vibration measurement is over the threshold described in a context definition. Figure 5.2 presents an illustration of the execution of a CsA.

Wait for the Initial Context (ES_1) Supporting business process execution engines of CsA activates them after the control flows in business processes reach the CsA. Each CsA has a goal defined by a goal definition, such as a goal of fixing a mobile device defect (G_1). Moreover, business experts associate each goal definition with context definitions describing the situation in which organizations need to achieve the goal described. By having machine-processable context definitions, business process executions of CsA can automatically detect the respective situations upon their presence using the available contextual data. As shown in Figure 5.2, each CsA holds the execution of this activity until the specified initial context is present in the environment (ES_1). Technical experts can additionally define a timer to limit the wait time. Upon a change in the contextual data, context definitions of activated CsA are evaluated. For example, the business process execution engine awaits the presence of mobile devices returned after completing their manufacturing process to execute a CsA aimed at fixing a mobile device defect.

Select Realization Mechanisms Based on the Current Context (ES_2) Following the presence of the context defined in a context definition indicating the initial context of a goal definition targeted by a CsA, the corresponding business process execution engine identifies every realization mechanism relevant for the present context (ES_2). Therefore, it iterates through each realization mechanism available in its repository. To this end, business experts create the set of realization mechanisms stored in each repository by defining their attributes, such as their goal definitions with initial context definitions. The business process execution engine evaluates the (initial) context definition of the goal definition targeted by each realization mechanism using the available contextual data. As a result, every realization mechanism targeting any goal with an initial context definition describing the current context is selected. To this end, "describing the current context" means that the context definition of a goal definition evaluates to true using the current contextual data. For example, a context definition specifying the presence of returned devices (e.g., returned devices > 0) will evaluate to true if the number of returned devices in the contextual data is higher than one (e.g., the number of returned devices is 30 in the contextual data).

As a result of step ES_2, there is a set of realization mechanisms targeting goals, whose initial context definitions describe the current context in the respective organization, as illustrated in Figure 5.2. For example, upon the presence of returned mobile devices after completing their manufacturing process, the corresponding business process engine of a CsA aimed at a goal of fixing a defect of the returned mobile device (G_1) selects (i) a semiautomated business process (RP_1) and (ii) a manual business process (RP_2) aimed at the same goal. Furthermore, business processes (RP_3 and RP_4) aimed at maintaining simulation machines (G_4) can be additionally selected depending on the values in the contextual data. Thus, context definitions referring to initial contexts of goal definitions of realization mechanisms play a critical role during this selection step, ES_2. As a result of not focusing on the goals but rather on their associated context definitions of realization mechanisms, it is possible that there are no realization mechanisms selected matching the goal of a CsA. In every case, there should exist at least one realization mechanism aimed at the goal in the initial context of a CsA. The absence of any realization mechanism after step ES_2 is a sign of an erroneous setup and requires a reconfiguration of the CsA and its realization mechanisms accordingly. Moreover, such an absence during the enactment of a CsA results in a runtime error. For example, if (i) a CsA aimed at fixing a mobile device defect is activated and (ii) no realization mechanisms target this goal, an error occurs.

Select Realization Mechanisms Based on Their Goals (ES_3) The resulting set of realization mechanisms after executing step ES_2 contains realization mechanisms aimed at possibly same goals (RP_1 and RP_2). Consequently, a selection among realization mechanisms aimed at the same goals must be done (ES_3). Selection strategies of goal definitions enable this selection. Each selection strategy take multiple realization mechanisms targeting the same goal; that is, they refer to the same goal definition. As a result, it returns a realization mechanism as output, as illustrated in Figure 5.2.

During ES_3, a realization mechanism aimed at the goal of a CsA is selected first. For example, a selection strategy of business processes aimed at investigating and fixing a mobile device defect can opt for a semiautomated business process due to the availability of simulation machines. In contrast, if the simulation machines are unavailable or defective, the selection strategy will select the manual version of the corresponding business process. To this end, a realization mechanism targeting the goal of a CsA is selected irreversibly, resulting in the elimination of every other realization mechanism aimed at any of the goals of the selected realization mechanism. Consequently, the selected realization mechanism aimed at the goal of a CsA cannot be eliminated afterwards by other selection strategies.

Hereafter, all overlapping goals (G_4) of remaining realization mechanisms are identified (RP_3 and RP_4). After the identification, the selection strategy of each overlapping goal is executed using (i) realization mechanisms aimed at the overlapping goals and (ii) the contextual data. Respective selection strategies can consider other goals during the selection of realization mechanisms. For example, a selection strategy can make a selection between two realization mechanisms with an overlapping goal based on the number of goals targeted by the realization mechanisms. Consequently, selection strategies adapt the business logic for achieving goals specified for CsA based on custom factors, such as the current context in an organization. If a goal definition does not have an associated selection strategy, the respective business process execution engine randomly selects one realization mechanism for every goal. Thus, exactly one realization mechanism targets the goal of CsA.

Optimize Selected Realization Mechanisms and Execute (ES_4 and ES_5) The final step before executing selected realization mechanisms is an optimization step (ES_4), as shown in Figure 5.2. For each realization mechanism referring to an optimization strategy, business process execution engines run the strategy to create a more optimal version of the corresponding business process for the current situation. Similar to selection strategies, optimization strategies rely on different factors, such as the contextual data in the organization, to optimize automatically a given realization mechanism. The results of executing optimization strategies are realization mechanisms with all the necessary information for executing the processes. After optimizing realization mechanisms, business process execution engines supporting CsA enact all realization mechanisms (ES_5). Therefore, the business process execution engines exploit other business process execution engines capable of executing realization mechanisms, such as a BPMN engine or a resource-driven process runtime. Notably, the input containers and output containers of CsA must match the selected realization mechanisms aimed at the goal of respective CsA, or each such realization mechanism should contain data mapping, enabling a mapping between these. Next, we discuss the new type of activity construct, CsA.

5.3 Discussion

The CsA refers to goal definitions specifying the initial context required to initialize activities. The concept of CsA enables modeling contexts in business processes and adaptations based on this. Thus, CsA satisfy the requirement Ri_1. Similarly, various work [Wie13; WKNL07; WKN08; WHR09; BLMP09; MGKR15; KBG+16; WRR08; GSBC00] addressed the modeling and execution of business processes based on contextual data.

Each CsA aims at accomplishing a goal by considering the current contextual data. As each context can require the execution of multiple business processes in parallel (Section 5.1.2), during the execution of CsA, every business process with a matching context definition is activated. To this end, a selection strategy of a goal definition ensures the selection of a business process capable of satisfying the goal specified by a CsA. Consequently, our approach satisfies Ri_2. Similarly, Mundbrod et al. [MGKR15] presented an approach enabling multiple business processes with matching context definitions in parallel. However, they did not present a goal orientation, as stated in Ri_2. Adaptive pervasive flows enable specifying different goals in terms of target context definitions. During execution, the necessary business logic resulting in the target context definition is created [BLMP09]. However, adaptive pervasive flows do not enable execution of context-sensitive goal-oriented multiple business processes simultaneously, as stated in Ri_2, but focus on creating business logic capable of reaching the target context. For example, a business process aimed at maintaining simulation machines will not be executed in an adaptive pervasive flow as long as it is not relevant for reaching a target context. In contrast, in case of CsA, such a business process can be additionally executed in parallel to another realization mechanism when its initial context definition describes the current context.

Notably, the CsA aims at combining business processes independently from their specification languages, such as BPEL, BPMN, and Redo. Therefore, realization mechanisms integrate different business process modeling languages and present the necessary information needed for initializing the business process described in the respective business process model. Thus, realization mechanisms enable the specification of business processes in terms of their goals in a unified way. Such a goal oriented definition of business processes can be observed in different approaches [FFST11; dSdSPvS09; GRG+04; STSJ08; MC10]. In addition to these different approaches, realization mechanisms specify goals with their respective selection strategies in a common format. Consequently, realization mechanisms refer to not only their goals but, also, the means of selecting among realization mechanisms aimed at the same goals.

For optimizing selected business processes before their execution, each realization mechanism can be associated with an optimization strategy. Before the execution of a realization mechanism, business process execution engines supporting CsA run associated optimization strategies. These optimization strategies can rely on the current contextual data for optimizing the realization mechanisms. Thus, the concept of CsA meets Ri_3. In different business process modeling and execution approaches, different types of optimization methods [HS15; HCMP16; BLMP09; WRR08; GSBC00; SL13; Mar+05; SPW+04; KLA16; KLA11] exist. By referring to optimization strategies explicitly in CsA, different optimization strategies targeted at different business process modeling languages can be employed. For example, an optimization strategy can rely on approaches that create an activity-oriented business process model based on planning [HS15]. Similarly, an optimization strategy can rely on algorithmic solutions aimed at optimal business process [Wib13; Ver08]. The concept of optimization strategies do not present a new way of optimizing business processes. Moreover, optimization strategies provide means of specifying which existing optimization approach should be used.

The CsA stands out by fulfilling all the requirements explained in Section 5.1 unlike the existing approaches explained in Section 2.3.1.1. To this end, the concept of realization mechanisms enables the association between an activity-oriented business process model and a resource-driven process. Furthermore, selection strategies provide means of selecting the matching realization mechanism depending on the current contextual data. Depending on the contextual data, multiple realization mechanisms can be executed for a CsA, as long as one realization mechanism aimed at the goal of the corresponding CsA exists. Finally, optimization strategies associated with realization mechanisms facilitate a more optimal enactment of realization mechanisms depending on the current context, if applicable. To validate CsA, we present an extension activity for BPMN in Section 6.3.

CHAPTER 6

EVALUATION AND VALIDATION OF CONCEPTS

This chapter presents the evaluation and validation of the concepts presented in this thesis. To validate and evaluate resource-driven processes presented in Chapter 3, we have conducted a series of user surveys. In Section 6.1, we present the summary of our results. To validate the Resource-driven Process Management (RPM) life cycle, we implemented a resource-driven process modeling, execution, and discovery environment. We detail our implementation in Section 6.2. To incorporate resource-driven processes into business processes modeled in an activity-oriented fashion, we presented Context-sensitive Activities (CsA) in Chapter 5. To validate CsA, we extended an open-source business process management suite. We present the details of our prototypical implementation supporting CsA in Section 6.3.

6.1 Empirical Analysis of Resource-driven Processes with Surveys

Resource-driven processes enable support for actors of business processes and the reproduction of desired outcomes of the business process, as explained in Section 3.1, by documenting and automatically allocating interrelated resources supporting these actors. These automatically allocated resources work collaboratively to reproduce desired outcomes of business processes. This section presents an empirical analysis of resource-driven processes regarding their support and reproducibility aspects based on a series of surveys. For the analysis, we opted for a quantitative method rather than a qualitative method, as we do not have any exploration purposes. Instead, we focus on validating and evaluating resource-driven processes based on the feedback of human actors of business processes. Next, we describe the setting of our quantitative research (Section 6.1.1) and present a summary (Section 6.1.2) and interpretation of the data (Section 6.1.3).

6.1.1 Empirical Evaluation

During the design of our quantitative research approach, we followed the steps proposed by Creswell [Cre13]. Therefore, we started by defining different variables, such as dependent, mediating, and independent variables, in our survey. In the context of our quantitative analysis, we use the definition of a variable from Creswell [Cre13]. Variables represent different characteristics and attributes of organizations or individuals varying among different people or organization being studied. Moreover, variables can be observed or measured [Cre13]. For example, (i) the degree of the provided support for actors during enactments of business processes and (ii) the reproducibility of desired outcomes of business processes represent observable characteristics of organizations. Thus, these characteristics are considered variables in our survey. More specifically, we consider the degree of the support provided for human actors of business processes as a continuous variable, and we aim at inferring (i) a positive or (ii) negative change of this variable in our survey.

Furthermore, the reproducibility of desired outcomes of business processes is considered a binomial categorical variable with categories true and false. Obviously, these variables are dependent variables, as they depend on other variables representing other characteristics of organizations. Interestingly, (i) the degree of support for actors during enactments of business processes and (ii) the reproducibility of desired outcomes of businesses can depend on documenting information regarding important resources used in a business process. Similarly, the automated allocation of organizational resources can (i) increase the degree of support for actors of business processes and (ii) enable the reproducibility of their desired outcomes. Furthermore, as the RPM life cycle facilitates (i) the documentation of important resources and (ii) the automated allocation of resources in organizations, the life cycle is a mediating variable binding a set of dependent variables and independent variables. Figure 6.1 illustrates a causality diagram representing the variables of our research based on the notation suggested by Duncan [Dun66]. The two-sided arrows show the correlations that are not investigated here. Moreover, one-sided arrows represent a dependency relationship between different variables.

The first phase (P_1) of the RPM life cycle aims at enabling the modeling and automated allocation of different categories of resources that are important for enactments of business processes, as explained in Section 4.3.1. For example, if IT resources are important for business processes of an organization, the RPM life cycle will enable documenting and automatically allocating different types of resources in this category by preparing modeling and execution environments of resource-driven processes during P_1. Thus, the importance of different categories of resources in organizations affects resources integrated during the first phase P_1 of the RPM life cycle. Moreover, different categories of resources can result in different degrees of support provided for actors in different organizations. Thus, identifying the overall degree of importance of different categories of resources in organizations will enable a judgment of the added value by documenting particular categories of resources, such as human, information, IT, and physical resources, as explained in Section 2.2. Another aspect to be considered in this work

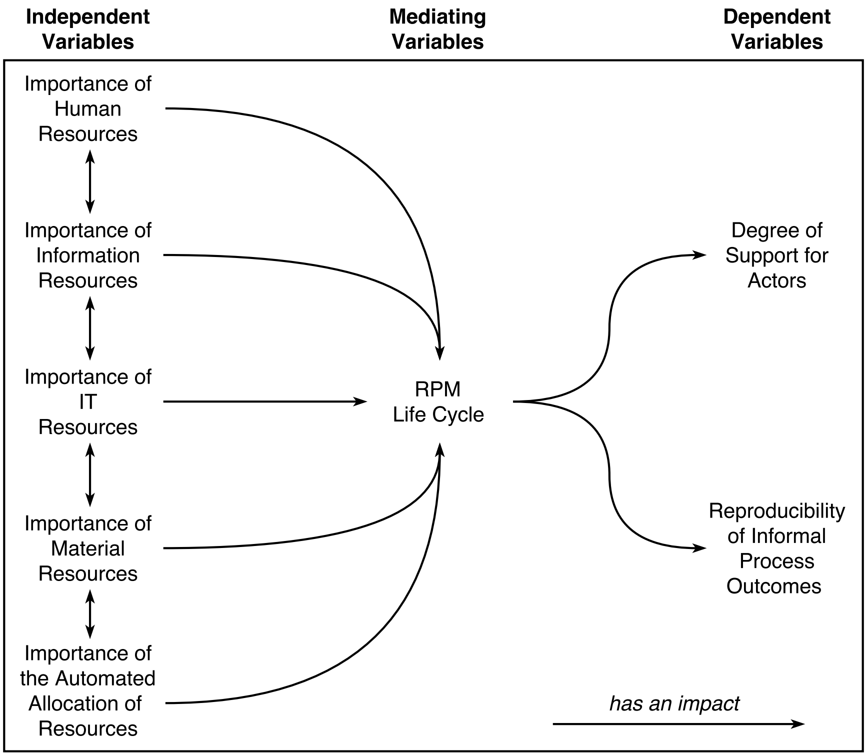

Independent Variables
Mediating Variables
Dependent Variables
Importance of Human Resources
Importance of Information Resources
Importance of IT Resources
Importance of Material Resources
Importance of the Automated Allocation of Resources
RPM Life Cycle
Degree of Support for Actors
Reproducibility of Informal Process Outcomes
has an impact

Figure 6.1: A causality graph between different variables of the survey.

is the added value of automatically allocating these resources. Therefore, in our survey, we investigate the importance of different categories of resources and the importance of the automated allocation of these to judge the added support using resource-driven processes. The importance of different categories of resources and the automated allocation of these in the survey are independent ordered categorical variables starting from the category not important (1) to very important (5) [SM96]. Finally, we investigate the reproducibility of desired outcomes of business processes in organizations by documenting and automatically allocating different categories of resources. In other words, we investigate the types of causality relationships presented in Figure 6.1, that is, impacts of an independent variable on dependent variables. For example, the higher importance of IT resources will result in an increased support through the RPM life cycle, as it enables the documentation and automated allocation of these resources.

To investigate the causality relationships between our variables illustrated in Figure 6.1, we used a survey method due to its economical design and rapid turnaround for data collection [Cre13]. Moreover, the data collected in our survey represent one point in time; we surveyed our participants for a period, not repeatedly. We collected data using web-based Internet surveys and self-administrated questionnaires to reach a larger population with low costs. Consequently, the data are primarily restricted to provided possible answers and do not allow exploration. We limited our study to people employed in Germany to isolate effects of different factors changing from one culture to another, such as the working culture of a country. Thus, the results represent employed people in Germany only. Moreover, the size of our target population (i.e., employees in Germany) is 43,056,000 according to 2015 numbers published by the Federal Office of Statistics of Germany [Bun16]. To calculate our sample size for the population, we used the equation presented by Cochran [Coc77; Isr92] with a margin of error of 5% and with a confidence level of 95%, as suggested by Bartlett et al. [BKH01] for categorical data. Using the equation, we identified the resulting sample size as 385. During the selection of our participants (samples) from our population, we identified researchers from different disciplines, as these researchers are typically involved in informal processes. Thus, they are good representatives of actors of business processes containing no activity structures. Moreover, we surveyed employees with different roles working in companies from different fields, such as telecommunications, mechanical engineering, and transportation in a job fair. Finally, we reached the majority of our participants over an online service, using random sampling. During this random sampling, we selected only those working in Germany with control questions. We composed a set of new questions addressing the importance of different categories of resources, the importance of automated allocation, and the reproducibility of desired outcomes of business processes using resources. More specifically, the questions discuss the importance of human, information, IT, and physical resources to generalize the need for each resource category. Further questions investigate the importance of the automated allocation of resources perceived by different participants to determine whether such an allocation is a desired feature of most participants to complete their daily tasks. Finally, a set of questions was created to investigate the reproducibility of business processes in their organizations based on resources with the same capabilities to confirm the reproducibility of desired outcomes of business processes. In other words, we asked a set of questions referring to independent variables in Figure 6.1 to analyze the effect of using resource-driven processes (i) on the degree of provided support for actors in business processes and (ii) the reproducibility of desired outcomes of business processes. The language of the survey was German for easier comprehension by the participants. Moreover, we specified questions using common language containing no terms specific to computer science, so that a common understanding of the questions from different disciplines is possible. Initially, we conducted the survey on a set of test subjects and improved our questions accordingly based on their feedback. Next, we present the results of the survey.

6.1.2 Results of the Survey

We collected data from 416 people employed in Germany, representing the statistically relevant sample size with a margin of error of 5% and with a confidence level of 95%. Figure 6.2 summarizes the answers to questions addressing (i) the importance of different categories of resources and (ii) the automated allocation of resources. Moreover, the results and respective calculations are presented online[1].The degree of importance is a categorical variable with the possible following values (1) not important, (2) optional, (3) undecided, (4) important, and (5) very important. Specifically, 146, 105,

[1]`https://github.com/timur-han/use-of-resources-in-organizations-survey`

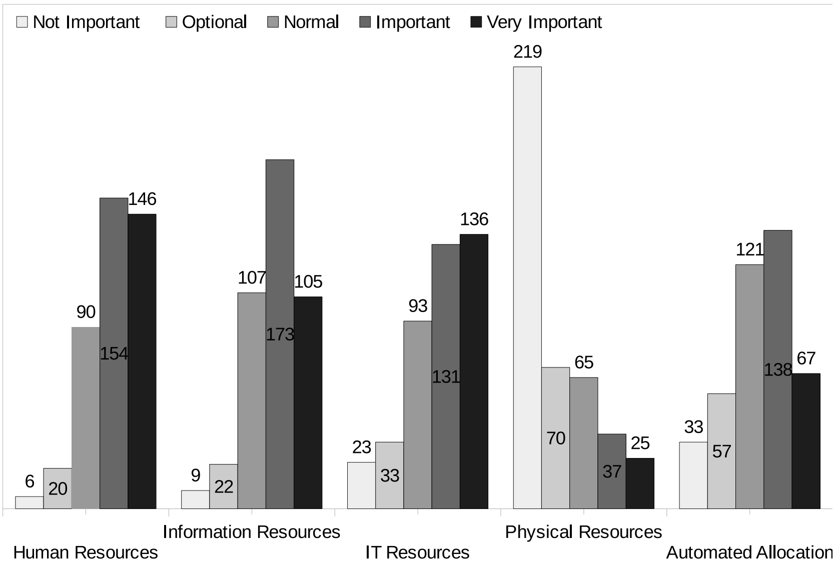


Figure 6.2: Importance of different categories of resources and the automated allocation of resources.

136, 25, and 67 of our participants thought that human, information, IT, and physical resources and the automated allocation of these resources are very important, respectively. Moreover, 154, 173, 131, 37, and 138 of our participants believed that human, information, IT, and physical resources and the automated allocation of resources are important, respectively. Surprisingly, 90, 107, 93, 65, and 121 of our participants were undecided about the degree of the importance of human, information, IT, and physical resources and the automated allocation of resources, respectively. In contrast, 20, 22, 33, 70, and 57 of our participants believed that human, information, IT, and physical resources and the automated allocation of resources are optional, respectively. Finally, 6, 9, 23, 219, and 33 of our participants believed that human, information, IT, and physical resources and the automated allocation of resources are not important, respectively. We analyzed each resource

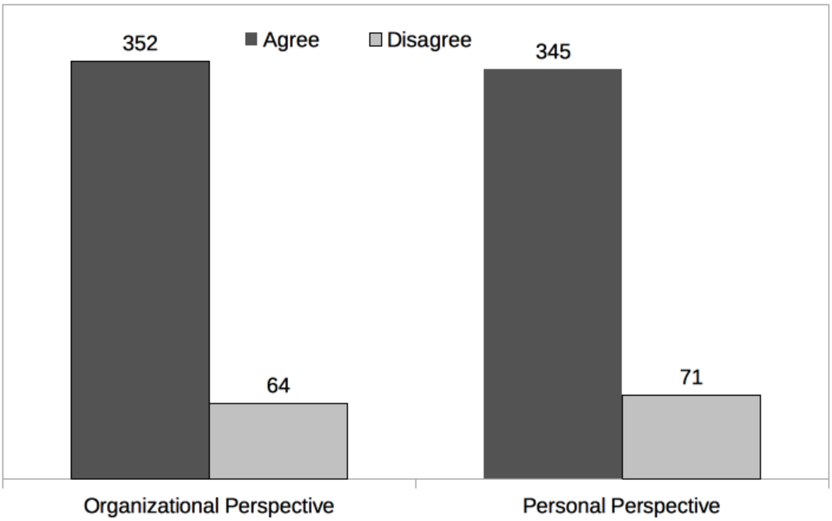


Figure 6.3: Opinions of participants about the reproducibility of desired outcomes using resources with the same capabilities.

category individually, and the average degrees of the importance of human, information, IT, and physical resources and the automated allocation of resources are 4, 3.82, 3.78, 1.99, and 3.36, respectively. Furthermore, the standard deviation in different datasets representing the importance of human, information, IT, and physical resources and the automated allocation of resources are 0.94, 0.94, 1.15, 1.26, and 1.14, respectively.

We asked participants about the reproducibility of desired outcomes of business processes using a set of resources. Figure 6.3 presents a summary of the results. To this end, we investigated two perspectives addressing (i) all business processes of an organization and (ii) those involving the participant being asked, only. For each perspective, we asked questions with possible answers of (1) agree and (2) disagree. Moreover, 352 and 345 of the participants agreed that the desired outcomes of all business processes of an organization and those involving the participant being asked can be reproduced using the resources with the same capabilities, respectively. In contrast, 64 and 71 of the participants disagreed that the desired outcomes of all business processes of an organization and those involving the participant being asked can be reproduced using the resources with the same capabilities, respectively.

We analyzed each perspective individually, and the averages of (1) agreeing and (2) disagreeing that the desired outcomes of all business processes of an organization and those involving the participant being asked can be reproduced are 1.15 and 1.17, respectively. Finally, the standard deviations of (1) agreeing and (2) disagreeing that the desired outcomes of all business processes of an organization and those involving the participant being asked can be reproduced are 0.34 and 0.38, respectively. For the sake of completeness, we provide a brief overview of correlation coefficients between investigated variables. Interestingly, correlation coefficients between most of the variables mean "little if any correlation" [ASG06]. In contrast, the correlation coefficient between the importance of human and information resources is 0.40, that is, "low correlation" (i.e., correlation coefficients are in the range of $(-0.3, +0.3)$ [ASG06]). Moreover, the correlation coefficient between (i) the reproducibility of desired outcomes of business processes involving participants and the importance of IT resources is -0.34, that is, "low correlation" [ASG06]. Because of meaning little if any correlation, we do not list the rest of the correlation coefficients between different variables. For further insights, we refer readers to the collected survey results provided in the online datasheet[1]. In the following section, we provide a discussion of the validation and evaluation of resource-driven processes using the RPM life cycle based on the survey data.

6.1.3 Discussion of the Results of the Survey

Based on the data in our survey, we can draw a set of conclusions about (i) the degree of support for actors and (ii) the reproducibility of desired outcomes of business processes enabled by resource-driven processes using the RPM life cycle. Moreover, we can evaluate resource-driven processes with other approaches aimed at supporting informal processes. Figure 6.4

presents a summary of our findings. In Figure 6.4, the numbers above the arrows present the averages of the importance of the respective categories of resources and the automated allocation of resources calculated using the survey data. Naturally, the more important a category of resource or automated allocation is, the higher effect it has on the RPM life cycle. For example, if an organization relies mainly on IT resources, technical experts make this category of resource available during the first phase of the RPM life cycle (P_1) by creating domain-specific information providers. Furthermore, depending on the importance of the automated allocation of resources, the corresponding domain-specific operation providers are created during P_1. The positive signs above the arrows represent a positive effect on dependent variables in Figure 6.4 caused by the importance of the independent variables. To this end, we use a positive or negative effect instead of certain numerical values, as the observation of such a quantitative change was impossible in the context of our survey. First, we focus on the support aspects of the results.

The IT resources are important (131) and very important (136) for most participants (64%), as shown in Figure 6.2. Moreover, physical resources are important (37) and very important (25) for some participants (15%). Although physical resources play a relatively less critical role for most participants in comparison with other categories, it is still important for a group of people, and documenting the reusable information regarding physical resources will increase the degree of support for the participants relying on physical resources. Furthermore, these numbers prove the relevance of IT and physical resources for increasing the degree of support for actors of business processes. Moreover, in addition to alternative approaches supporting knowledge workers, such as content-oriented approaches (Section 2.3.3), resource-driven processes enable the reuse of the information regarding physical resources.

Another aspect that is typically not addressed by other approaches is the automated allocation of resources required. We believe the relatively high number of undecided participants (121) regarding the automated allocation of resources is a result of the newness of this idea. As illustrated in Figure 6.2, the majority of participants excluding the undecided ones (69%) ranked

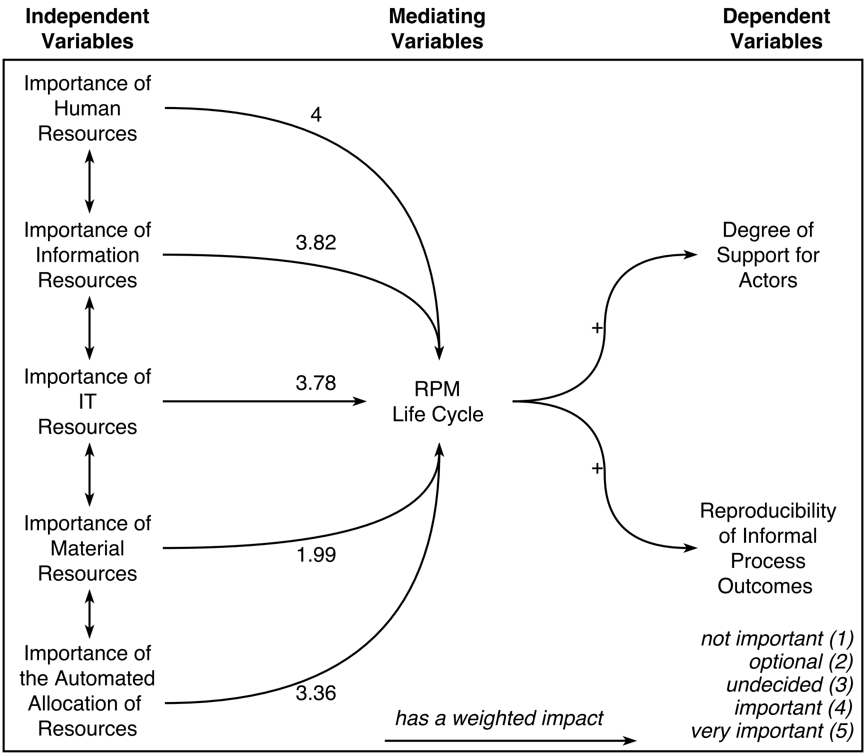

Independent Variables
Mediating Variables
Dependent Variables
Importance of Human Resources
Importance of Information Resources
Importance of IT Resources
Importance of Material Resources
Importance of the Automated Allocation of Resources
4
3.82
3.78
1.99
3.36
RPM Life Cycle
+
+
Degree of Support for Actors
Reproducibility of Informal Process Outcomes
not important (1)
optional (2)
undecided (3)
important (4)
very important (5)
has a weighted impact

Figure 6.4: An overview of the results of the survey.

the automated allocation of resources as important and very important. Thus, by enabling the automated allocation of resources using the RPM life cycle, organizations will increase the support provided for actors of business processes. Using the RPM life cycle in organizations enables (i) documenting the investigated important resources of business processes and (ii) automating the allocation of the documented resources. Thus, the RPM life cycle influences the degree of support for actors positively.

Finally, the results of questions regarding the reproducibility of desired outcomes of business processes show that most believed that the desired outcomes of all business processes of an organization and those involving the participant being asked can be reproduced based on resources providing the same capabilities. Thus, the RPM life cycle influences the reproducibility of desired outcomes of business processes positively by automatically allocating important resources with certain capabilities for business processes. The consensus on the reproducibility of desired outcomes of business processes is an expected result, as various organizations employ certain resources providing certain capabilities for enacting their business processes. For example, an organization can employ the same developer for fixing a mobile device defect due to his or her familiarity with the mobile device to reproduce desired outcomes, such as a fixed mobile device. In summary, the data provide empirical evidence supporting our claims about (i) increasing support for human actors of business processes and (ii) reproducing their outcomes using resource-driven processes (Section 1.2.1). Please note that a set of responses regarding the importance of human resources can be biased, as participants themselves are human resources. Consequently, a set of participants may have assigned a higher value of importance to human resources, although they are actually not so important for activities of the participants. During the interpretation of the given results, this fact must be taken into consideration. Next, we present the validation of the RPM life cycle.

6.2 Validation of the RPM Life Cycle

To validate the RPM life cycle, we created a prototypical implementation[2] of a resource-driven process modeling environment (Section 4.2.1), a resource-driven process execution environment (Section 4.2.2), and a resource-driven process discovery environment (Section 4.2.3). As detailed in Figure 4.2, a resource-driven process modeling environment is composed of a set of resource-driven modeling tools capable of creating and updating different elements of the resource-driven process modeling language and at least

[2]`https://bitbucket.com/timurhan7/ipsm`

one domain-specific information provider, as explained in Definition 33. In our prototype, we used enterprise integration patterns by Hohpe and Woolf [HW04]. For example, we created domain-specific information and operation providers in a loosely coupled way using message queues. The messages sent by domain-specific information and operation providers are consumed by resource-driven process modeling and execution environments in an event-driven way. Consequently, each domain-specific information and operation provider can join and leave the system in a loosely coupled fashion. To ease the development of domain-specific information and operation providers, we created a client library hiding the details of the communication between domain-specific information and operation providers and organizational stores in Java. As a consequence of using this library, technical experts can focus on the development of domain-specific functionality of each domain-specific information and operation provider and leave the communication details to the library itself.

In our prototypical implementation, we focused on the human, IT, and information resources due to the relatively high importance of these categories of resources in our survey. Integrating physical resources can be done similarly using IT services capable of providing information and controlling these resources, such as enterprise resource planning services [OLe00]. To represent resource (Definition 21) and relationship types (Definition 23), we used the node and relationship types of the TOSCA specification (Section 2.4.1). Relying on the TOSCA specification facilitated the reuse of tools built for TOSCA, such as Winery for managing resource and relationship types. For collecting information regarding available human resources in organizations, we developed a domain-specific information provider using the open-source social networking service Elgg [Sha08]. Similarly, we created a domain-specific operation provider of human resources capable of allocating human resources by interacting with them using the social networking service.

To include the IT resources in our resource-driven process modeling and execution environment, we developed a domain-specific information provider and domain-specific operation provider, exploiting Docker and Docker Com-

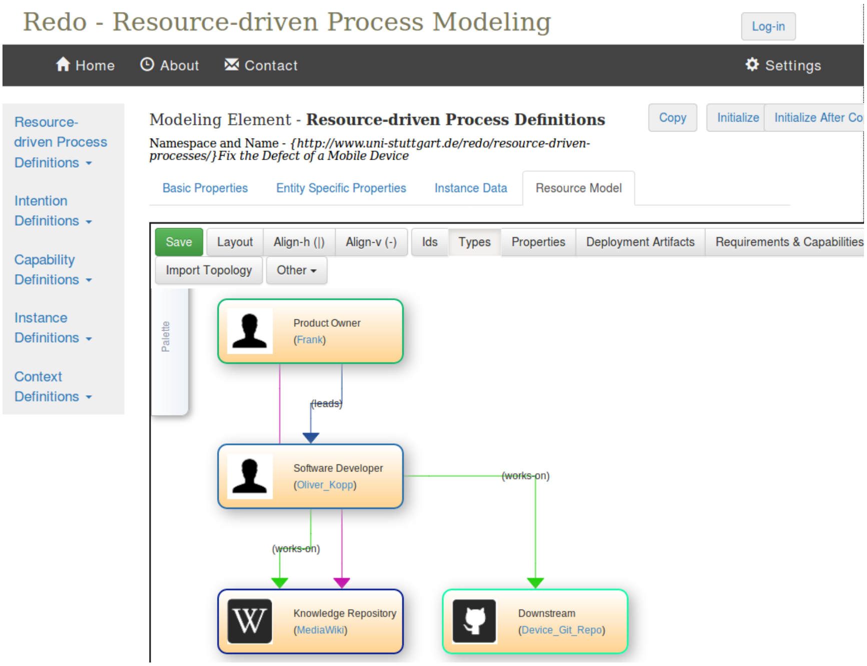

Redo - Resource-driven Process Modeling
Log-in
Home
About
Contact
Settings
Resource-driven Process Definitions
Intention Definitions
Capability Definitions
Instance Definitions
Context Definitions
Modeling Element - Resource-driven Process Definitions
Copy
Initialize
Initialize After Co
Namespace and Name - {http://www.uni-stuttgart.de/redo/resource-driven-processes/}Fix the Defect of a Mobile Device
Basic Properties
Entity Specific Properties
Instance Data
Resource Model
Save
Layout
Align-h (|)
Align-v (-)
Ids
Types
Properties
Deployment Artifacts
Requirements & Capabilities
Import Topology
Other
Palette
Product Owner (Frank)
(leads)
Software Developer (Oliver_Kopp)
(works-on)
(works-on)
Knowledge Repository (MediaWiki)
Downstream (Device_Git_Repo)

Figure 6.5: Screenshot of a web-based resource-driven process modeling tool.

pose (Section 2.4.1). The knowledge disseminated during enactments of business processes is typically important for the future enactments of these. Therefore, information resources store the explicit knowledge disseminated during enactments of business processes and other data relevant to resources, such as a Wiki service allocated for business processes and a database containing test results of a software. To include such information resources, we created a domain-specific information provider and a domain-specific operation provider of information resources, such as documents, a Wiki service, an SQL database, tables in a relational database, and other resources containing explicit knowledge regarding business processes [Wed16]. Upon registration, each domain-specific information provider sends its available resource and relationship types to corresponding resource-driven process modeling and execution environments over a message channel, and the algorithm presented in Algorithm 1 is executed by our resource-driven process modeling environment.

We developed a backend service capable of creating, updating, retrieving, and deleting context definitions, goal definitions, capability definitions, and resource-driven process definitions. The backend provides a REST API for its clients. Moreover, we created two resource-driven process modeling tools (Definition 31) to carry out the steps presented in Section 4.2.1. The browser client is an HTML 5 application relying on the topology modeler of Winery to create resource models of resource-driven process definitions [Kal16]. Figure 6.5 presents a screenshot of this web-based modeling tool. During enactments of business processes, there can be the need for (i) ad hoc changes in resource-driven process definitions and (ii) access to information regarding resource-driven processes on the go [KT16]. Therefore, we created a mobile client for business experts, enabling the management of resource-driven process definitions from Android devices [Che16]. Moreover, we developed a resource-driven process execution environment (Definition 33) for initializing the created resource-driven process definitions, as explained in Section 4.3.3. Based on the interactions created during enactments of business processes, resource-driven process discovery environments (Definition 54) generate recommendations to support business experts for modeling resource-driven processes (P_1), as explained in Section 4.3.4.

To validate the generation of relevance relationships containing relevant resources and relevant capabilities and new resource-driven process definitions containing relevant resource and capabilities, we created a case study around the software development project Apache jclouds [Son16a]. In our case study, we focused on GitHub interactions of the project to draw conclusions about the relevance of certain resources and capabilities. Therefore, we created two resource analyzers capable of collecting interaction analyses (Definition 44) of human resources and GitHub repositories, such as interaction analyses representing Git commits, watching a repository in GitHub, and creating issues, pull requests, and reviews to pull requests made.

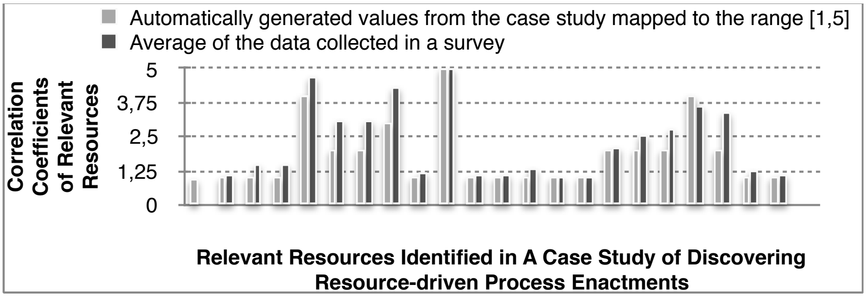


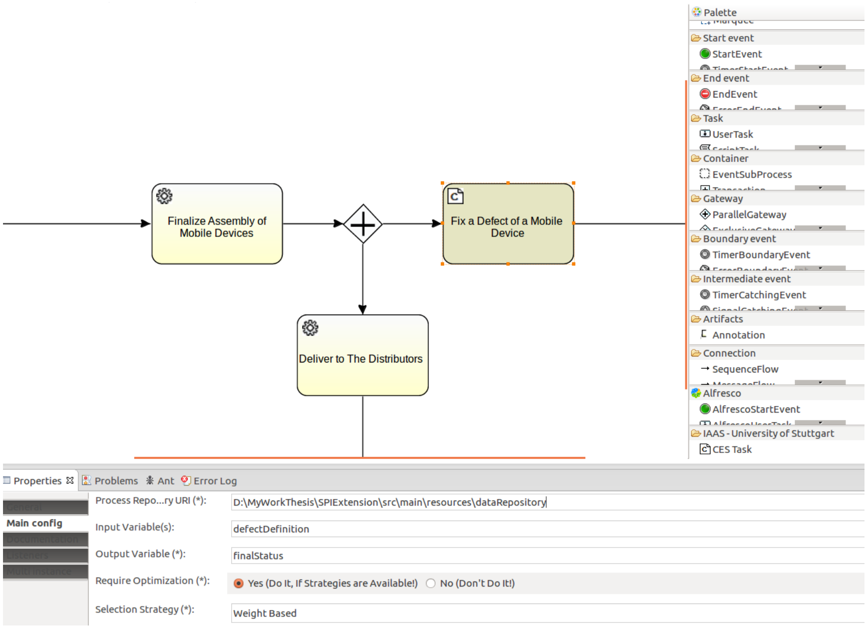
Figure 6.6: Comparison of automatically generated correlation coefficients with survey data.

To generate relevance relationships out of collected interaction analyses, we use three relevance mappers. To this end, one of the relevance mappers is capable of generating relevance relationships of relevant resources out of interaction analyses. For example, it deduces a relevance relationship for a software developer based on the commit interaction with a Git repository.

Similarly, the second relevance mapper uses interaction analyses to identify a set of relevance relationships of capabilities, such as a relevance relationship containing a software development capability generated based on a commit interaction. In addition to relevance mappers relying on interaction analyses, we created a third relevance mapper using existing relevance relationships to generate new relevance relationships of capabilities or update existing ones. For example, such a relevance mapper will use a relevance relationship containing a MediaWiki service to generate a relevance relationship containing a knowledge base capability. Lastly, we created a threshold-based resource-driven process definition recommender (Definition 51) to generate a new resource-driven process definition containing relevant resources with a certain degree of relevance specified by a threshold value. To evaluate the equation presented in Definition 55, we conducted a survey with 13 GitHub experts using GitHub from 6 months to 72 months with an average of 34 months. The results of this survey and the respective calculations can be found in our respective Git repository[3]. We presented the experts with 21 relevant resources and their respective 1069 interactions in eight different types, which were identified during our case study on Apache jclouds. We expected each participant to assign a degree of relevance between 1 (lowest) and 5 (highest) to each resource after considering the interactions involving these resources. After collecting the expert opinions about the relevance of resources, we compared these with automatically generated correlation coefficients. Therefore, we mapped the range of the automatically generated correlation coefficients to values from 1 to 5. Figure 6.6 shows the comparison of the collected results and the automatically generated results after this mapping.

The generated values have a very high correlation (0.93) [ASG06] with the average of the expert opinions. More specifically, the correlation between the automatically generated values and individual opinions of GitHub experts vary between 0.79 and 0.98. Furthermore, our resource-driven discovery environment can estimate the values within 18% of a mean absolute percentage error [Arm78]. One of the reasons for the differences between the results is the higher precision of automatically generated correlation coefficients. In contrast, the correlation coefficients assigned by GitHub experts are integers ranging from 1 to 5 resulting in lower precision. Moreover, the equation (Definition 55) uses relevance factors representing the rate of the change in the correlation coefficient for every interaction analysis and relevance relationship. In our case study, relevance factors did not meet the agreement of participants precisely, which can be changed by updating the configuration of relevance factors during P_1, as explained in Section 4.3.1.

[3]`https://bitbucket.com/timurhan7/informal-process-recommender`



Process Repo...ry URI (*): D:\MyWorkThesis\SPIExtension\src\main\resources\dataRepository

Input Variable(s): defectDefinition

Output Variable (*): finalStatus

Require Optimization (*): [✓ Yes (Do It, If Strategies are Available!)] No (Don't Do It!)

Selection Strategy (*): Weight Based

Figure 6.7: Screenshot of the extended Activiti Process Designer.

6.3 Validation of Incorporating Resource-driven Processes into Activity-oriented Business Processes

To incorporate resource-driven processes into activity-oriented business process models, we introduced a new type of activity called CsA, as detailed in Chapter 5. To validate the application of CsA, we extended the open-source business process management suite Activiti to enable modeling and executing business process models containing CsA constructs [Rad12]. Our prototypical implementation[4] enables modeling business processes containing CsA constructs by extending BPMN, as illustrated in Figure 6.7. To this end, the availability of a database containing realization mechanisms is

[4]https://bitbucket.com/timurhan7/context-sensitive-manufacturing-processes

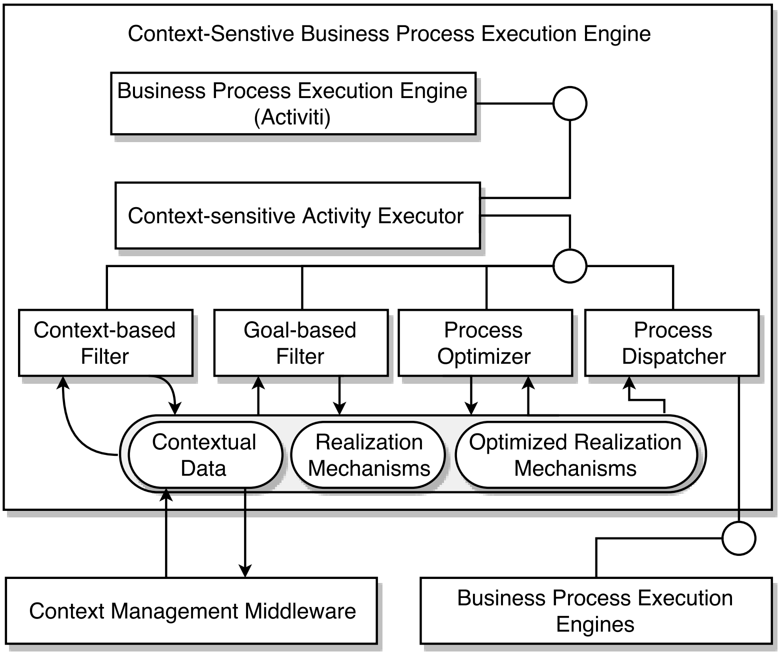

Context-Senstive Business Process Execution Engine
Business Process Execution Engine (Activiti)
Context-sensitive Activity Executor
Context-based Filter
Goal-based Filter
Process Optimizer
Process Dispatcher
Contextual Data
Realization Mechanisms
Optimized Realization Mechanisms
Context Management Middleware
Business Process Execution Engines

Figure 6.8: An overview of the architecture of the CsA execution environment using FMC notation [KW03; KGT06].

needed and must be specified for each CsA construct. Moreover, business experts specify the goal aimed by CsA constructs and corresponding selection strategies of the respective goal. To implement the execution semantics explained in Figure 5.2, we followed a pipes-and-filters architectural style relying on the enterprise integration patterns of Hohpe and Woolf [HW04]. More specifically, we created different filters, such as a context-based filter, as shown in Figure 6.8. In our prototype, we used already defined components of Apache Camel framework [IA10], implementing the enterprise integration patterns.

As illustrated in Figure 6.8, a context management middleware (Section 2.3.1.1), such as Nexus platform [LCG+09], delivers contextual data. To this end, the context-based filter updates the set of realization mechanisms based on the available contextual data, as explained in ES_2 in Section 5.2. After a message with the set of available realization mechanisms is passed to the context-based filter, it selects only realization mechanisms with context definitions, describing the current context in the organization. Similarly, the goal-based filter ensures that, for each goal, only one realization mechanism exists after executing ES_3. Context-based and goal-based filters are content filters from the enterprise integration patterns. Moreover, these filters are implemented using processor objects in Apache Camel.

The filtered realization mechanisms are optimized in the process optimizer component during ES_4. Process optimizer runs the associated optimization strategy of realization mechanisms and returns the optimized realization mechanisms. Lastly, the process dispatcher uses external business process execution engines to dispatch business processes described by realization mechanisms, such as a resource-driven process execution environment.

To test our prototypical implementation, we created a case study [Kar16] based on a manufacturing scenario presented by Erlach [Erl13]. Specifically, the manufacturing scenario focuses on the production of electronic blankets used mainly in Southern European countries. In this scenario, we converted a sub-process relying on contextual data into a CsA, aimed at packing and palletizing. Moreover, we specified (i) four realization mechanisms aimed at packing and palletizing, (ii) a realization mechanism aimed at maintaining machines used during packing and palletizing, and (iii) a business process, optimizing the execution of a realization mechanism. To this end, realization mechanisms targeting packing and palletizing are selected based on different criteria during the execution, such as different number of orders, available workers, and availability of machines. Furthermore, the realization mechanism aimed at maintaining machines is activated when a machine failure has been detected during the predictive maintenance analysis. Finally, the business process aimed at the optimization is enacted when a manual alternative of realization mechanisms aimed at packing and palletizing is selected. Hereafter, we validated the execution semantics of the CsA explained in Figure 5.2 with different contextual data.

Chapter 7

Conclusion and Outlook

This thesis introduces the concept of resource-driven processes for (i) supporting business processes and (ii) reproducing their desired outcomes without explicitly specifying their interrelated activities. To specify resource-driven processes, we introduced the formal language Resource-driven Process Modeling Language (Redo). Definitions of resource-driven processes created in Redo enable the documentation of interrelated resources to reach organizational goals. Furthermore, definitions of resource-driven processes are initializable. The initialization of resource-driven processes results in the automated allocation of interrelated resources required to reproduce desired outcomes of business processes.

To evaluate and validate the resource-driven approach, we conducted a survey of 416 people. The results of the survey showed that a generic resource concept capable of (i) representing different categories of resources, such as IT and physical resources and (ii) automating the allocation of resources increases the degree of support provided for human actors. Moreover, most participants think that business processes can be repeated using resources with equivalent capabilities, which supports our claim of reproducing desired outcomes of business processes using resource-driven processes.

To use resource-driven processes in organizations, we presented a Resource-Driven Process Management (RPM) life cycle containing four different phases. The first phase of our life cycle contains steps for identifying interrelated resources of business processes to configure their IT environments appropriately. More precisely, technical experts create necessary extensions for their IT infrastructure enabling (i) modeling resource-driven processes, (ii) automating the allocation of interrelated resources of resource-driven processes, and (iii) analyzing historical interactions of enactments of business processes to generate recommendations for business experts at modeling time. Using the prepared IT infrastructure, business experts can model and initialize definitions of resource-driven processes in an ad hoc fashion. Consequently, business experts can model and initialize resource-driven processes to handle unanticipated situations reactively.

After completing the first phase, business experts proceed with the modeling phase of resource-driven processes. Modeling starts with creating organizational goals and continues with associating necessary capabilities with these goals. Furthermore, business experts resolve these goals into their interrelated resources in definitions of resource-driven processes. To this end, required capabilities specified for goals support business experts during the selection of resources for accomplishing the goals. Definitions of resource-driven processes are then initialized in the next phase of our life cycle. Upon the initialization of resource-driven processes, interrelated resources marked as required are allocated automatically by executing corresponding allocation operations, such as allocating a Wiki service in an IT infrastructure. Thereafter, allocated resources collaboratively work toward goals of business processes. Thus, different allocated resources provide support for each other during the enactments of resource-driven processes.

A natural result of these collaborations is interactions residing in different types of interaction logs, such as a Git repository holding information on changes in the source code. Moreover, analyzing these interactions can enable conclusions about executed resource-driven processes. For example, by analyzing Git interactions in a development process, Git repositories and developers that are important and unimportant for such business processes can be identified. Interestingly, showing important and unimportant resources to business experts at modeling time of resource-driven processes can support their decision-making processes. Furthermore, including important resources and excluding unimportant resources from future enactments can result in better performance of business processes. Therefore, in the last phase of the life cycle, interactions between resources involved in business processes are automatically analyzed to generate resource-centric recommendations, which are recommended and nonrecommended resources, their capabilities, and resource-driven processes containing these. The first phase of the life cycle is repeated to adapt the organization to new business requirements. After the first phase, business experts continue to adapt existing models using generated recommendations and create new definitions of resource-driven processes. To validate our life cycle, we created a prototypical implementation of a resource-driven process modeling, execution, and discovery environment. Furthermore, we conducted a user survey to evaluate our recommendation method, proving its usability with the right configuration.

Finally, we present a new type of activity called Context-sensitive Activities (CsA) for incorporating resource-driven processes into activity-oriented business process models, which contain activities of business processes and the order of these activities. This new activity enables the selection of a business logic achieving a certain goal based on the current situation (context). For example, it enables the selection of business logic aimed at fixing a mobile device defect in the presence of the returned mobile devices. During the execution of CsA, multiple business processes can be initialized in parallel as long as exactly one business process aimed at the goal of the CsA exists. For example, a CsA aimed at fixing a mobile device defect and relying on simulation machines can additionally initialize in parallel a resource-driven process to maintain the simulation machines depending on the current context. Before executing the business processes selected during the enactment of a CsA, the business processes are optimized, if such automated optimization measures are available. To validate CsA, we extended a business process management suite. As a result, we can create business process models containing CsA, and we can execute these.

In future work, we will investigate the quality-of-service (QoS) aspects of resource-driven processes. As the quality of resource-driven processes will typically depend on allocated resources, it will be useful to extend definitions of resources with their QoS aspects. Furthermore, resources can be allocated by different providers with different QoS metrics. For example, a software developer can be allocated using internal human resources or using a service contractor with different fees per hour. Making these QoS aspects of resources transparent will support the decision-making process of business experts when modeling resource-driven processes. Consequently, business experts will be able to determine more adequate resources, and the resulting resource-driven processes will perform better. To this end, the WS-Policy standard and Policy4TOSCA are used to specify the QoS aspects of web services and application components, respectively [WCL+05; Wai+13]; thus, they present relevant concepts for starting the investigation of the quality-of-service for resources in resource-driven processes.

Goals achieved during resource-driven processes are typically accomplished in the context of higher-level goals. Furthermore, organizations achieve their goals during executions of their organizational strategies. Connecting resource-driven processes to their higher-level goals and to their strategies will increase the transparency in organizations. Moreover, the progress of organizational strategies will be traceable in terms of the completion of recursive goals resolving into resource-driven processes. Interestingly, with the aforementioned QoS aspects of resources, certain estimations, such as cost estimations of strategies, will be possible. Therefore, we will additionally extend Redo with higher-level goals and strategies for increasing transparency in organizations. Such declarative strategy-oriented process definitions have been proposed by Nurcan et al. [NEK+05], presenting a starting point for such an investigation. Finally, technological advancements in autonomous driving [Flä15] enables new prospects in the automated allocation of physical resources, which we will investigate in future work.

LIST OF FIGURES

List of Symbols

ad	An initializable entity definition $ad \in AD$ (Definition 17)
AD	The set of initializable entity definitions (Definition 17)
c	A capability definition $c \in \mathcal{C}$ (Definition 22)
$\mathcal{C}$	The set of capability definitions (Definition 22)
ct	A context definition $ct \in CT$ (Definition 27)
CT	The set of context definitions (Definition 27)
de	The set of resource-driven process discovery environments (Definition 54)
DE	A resource-driven process discovery environment $de \in DE$ (Definition 54)
dip	A domain-specific information provider $dip \in DIP$ (Definition 30)
DIP	The set of domain-specific information providers (Definition 30)
dom	Domain function is a function mapping from a given multivariable function and an index to the domain set of the function identified by the given index (Definition 10)
dop	A domain-specific operation provider $dop \in DOP$ (Definition 37)
DOP	The set of domain-specific operation providers (Definition 37)
ed	An entity definition (Definition 13)
ED	The set of entity definitions $ed \in ED$ (Definition 13)
es	A resource-driven process execution environment $es \in ES$ (Defini-

	tion 42)
ES	The set of resource-driven process execution environments (Definition 42)
g	A goal definition $g \in \mathcal{G}$ (Definition 28)
$\mathcal{G}$	The set of goal definitions (Definition 28)
gen_{IA}	Interaction analysis generator is a function mapping from an interaction collector and an instance descriptor, representing a resource definition to a set of interaction analyses (Definition 46)
gen_{RR}	Relevance relationship generator is a function mapping from a relevance mapper, a set of relevance relationships, and a set of interaction analyses to a set of relevance relationships (Definition 50)
$gen_{\mathcal{P}}$	Resource-driven process definition generator is a function mapping from a resource-driven process definition recommender, a resource-driven process definition, and a set of relevance relationships to a resource-driven process definition (Definition 52)
gen_{CC}	Correlation coefficient calculator is a function mapping from (i) an entity identity representing a resource definition or a capability definition and (ii) a set of interaction analyses or relevance relationships to a correlation coefficient (Definition 55)
img	Range function is a function mapping from a function to its range (Definition 11)
i	An instance descriptor $i \in \mathcal{I}$ (Definition 16)
$\mathcal{I}$	The set of instance descriptors (Definition 16)
ia	An interaction analysis $ia \in IA$ (Definition 44)
IA	The set of interaction analyses (Definition 44)
ic	An interaction collector $ic \in IC$ (Definition 43)
IC	The set of interaction collector (Definition 43)
id	An entity identity (Definition 14)
ID	The set of entity identities $id \in ID$ (Definition 14)
$ID_{lifecycle}$	The set of life cycle operation types (Definition 35)
$id_{lifecycle}$	A life cycle operation type $id_{lifecycle} \in ID_{lifecycle}$ (Definition 35)
ie	An identifiable entity definition $ie \in IE$ (Definition 15)
IE	The set of identifiable entity definitions (Definition 15)

me	A modeling element $me \in ME$ (Definition 18)
ME	The set of modeling elements (Definition 18)
ms	A resource-driven process modeling environment $ms \in MS$ (Definition 33)
MS	The set of resource-driven process modeling environments (Definition 33)
OD	The set of organizational definitions (Definition 9)
od	An organizational definition $od \in OD$ (Definition 9)
oe	An operation endpoint $oe \in OE$ (Definition 20)
OE	The set of operation endpoints (Definition 20)
oeb	An operation binding $oeb \in OEB$ (Definition 34)
OEB	The set of operation bindings (Definition 34)
$oe_{lifecycle}$	A life cycle operation endpoint $oe_{lifecycle} \in OE_{lifecycle}$ (Definition 36)
$OE_{lifecycle}$	The set of life cycle operation endpoints (Definition 36)
p	A resource-driven process definition $p \in \mathcal{P}$ (Definition 29)
$\mathcal{P}$	The set of resource-driven process definitions (Definition 29)
π	Projection function is a function mapping from a tuple and an index to the element at that index (Definition 12)
pc	A resource-driven process runtime $pc \in \mathcal{P}C$ (Definition 38)
$\mathcal{P}C$	The set of resource-driven process runtimes (Definition 38)
pd	A property definition $pd \in PD$ (Definition 19)
PD	The set of property definitions (Definition 19)
pm	A resource-driven process modeling tools $pm \in \mathcal{P}M$ (Definition 31)
$\mathcal{P}M$	The set of resource-driven process modeling tools (Definition 31)
po	The set of resource-driven process discoverers (Definition 53)
$\mathcal{P}O$	A resource-driven process discoverer $po \in \mathcal{P}O$ (Definition 53)
$\wp(S)$	Power set of the set S
pr	A resource-driven process definition recommender $pr \in \mathcal{P}R$ (Definition 51)
$\mathcal{P}R$	The set of resource-driven process definition recommenders (Definition 51)
r	A relationship type $r \in \mathcal{R}$ (Definition 23)
$\mathcal{R}$	The set of relationship types (Definition 23)

rd	A relationship definition $rd \in RD$ (Definition 25)
RD	The set of relationship definitions (Definition 25)
rm	A resource model $rm \in RM$ (Definition 26)
RM	The set of resource models (Definition 26)
rp	A relevance mapper $rp \in RP$ (Definition 49)
RP	The set of relevance mappers (Definition 49)
rr	A relevance relationship $rr \in RR$ (Definition 48)
RR	The set of relevance relationships (Definition 48)
s	A resource type $s \in \mathcal{S}$ (Definition 21)
$\mathcal{S}$	The set of resource types (Definition 21)
sd	A resource definition $sd \in SD$ (Definition 24)
SD	The set of resource definitions (Definition 24)
σ_{OE}	Operation endpoint selector is a function mapping from a resource type and an operation type referred to by two entity identities to an operation endpoint supporting the resource type in the given operation type (Definition 39)
σ_{RD}	Relationship retriever is a function mapping from a resource definition and a resource model to a set of relationship definitions indicating the resource definition as their source and target resource definition (Definition 40)
$\sigma_{\mathcal{I}}$	Child instance descriptor retriever is a function mapping from an entity identity referring to a resource, relationship, or goal definition and an entity identity indicating a parent instance descriptor to a child instance descriptor representing an instance of the modeling element referred to by the first parameter and initialized in the context of an instance described by the second parameter (Definition 41)
σ_{ME}	Entity identity resolver is a function mapping from an entity identity of a modeling element to the modeling element (Definition 32)

ACKNOWLEDGEMENTS

Finally, I want to thank all who supported me during my Ph.D process. Most importantly, I am grateful to Prof. Dr. Dr. h.c. Frank Leymann for the opportunity of doing my Ph.D. under his supervision and for sharing his valuable experience and vision with me through this process. Secondly, I want to express my gratitude to Univ. Prof. Dr. Schahram Dustdar for his detailed review of the thesis and to Prof. Dr.-Ing. habil. Bernhard Mitschang for his valuable feedback to my work during my presentations. Moreover, I thank my colleagues for their productive collaborations and resourceful discussions, which increased the quality of this work. Specifically, collaborations with Uwe Breitenbücher, Kálmán Képes, Oliver Kopp, Michael Hahn, Karolina Vukojevic-Haupt, and Andreas Weiß have increased the quality of this work significantly. Finally, I want to thank my friends and parents, Sibel Sungur and Gökhan Sungur, for standing by me all this time. Without their support, I could not come so far.

Bibliography

[ADG09] R. Ali, F. Dalpiaz, and P. Giorgini. “A Goal Modeling Framework for Self-contextualizable Software.” In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, June 2009, pp. 326–338 (cit. on p. 33).

[AGL98] R. Agrawal, D. Gunopulos, and F. Leymann. “Mining Process Models from Workflow Logs.” In: *Advances in Database Technology — EDBT’98*. Springer, 1998, pp. 467–483 (cit. on p. 37).

[AIM10] L. Atzori, A. Iera, and G. Morabito. “The Internet of Things: A survey.” In: *Computer Networks* 54.15 (Oct. 2010), 2787–2805 (cit. on p. 44).

[AMP94] A. I. Antón, W. M. McCracken, and C. Potts. “Goal decomposition and scenario analysis in business process reengineering.” In: *Advanced Information Systems Engineering: CAiSE’94*. Springer, 1994, pp. 94–104 (cit. on p. 33).

[Ant96] A. I. Anton. “Goal-based requirements analysis.” In: *Proceedings of the Second International Conference on Requirements Engineering (RE’ 96)*. IEEE, Apr. 1996, pp. 136–144 (cit. on p. 33).

[Arm78] J. S. Armstrong. *Long-range Forecasting: From Crystal Ball to Computer*. 2nd ed. John Wiley & Sons Inc, 1978 (cit. on p. 231).

[AS93] R. Amit and P. J. H. Schoemaker. “Strategic assets and organizational rent.” In: *Strategic Management Journal* 14.1 (1993), pp. 33–46 (cit. on p. 32).

[ASG06] A. G. Asuero, A. Sayago, and A. G. González. “The Correlation Coefficient: An Overview.” In: *Critical Reviews in Analytical Chemistry* 36.1 (2006), pp. 41–59 (cit. on pp. 223, 231).

[ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. “Wireless sensor networks: a survey.” In: *Computer Networks* 38.4 (Mar. 2002), pp. 393–422 (cit. on p. 45).

[AtHRvdA09] M. Adams, A. ter Hofstede, N. Russell, and W. M. P. van der Aalst. “Dynamic and Context-Aware Process Adaptation.” In: *Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments: Techniques for Adaptability in Turbulent Environments*. IGI Global, 2009. Chap. 5, pp. 104–137 (cit. on pp. 44, 46).

[BAdR06] K. Balog, L. Azzopardi, and M. de Rijke. “Formal Models for Expert Finding in Enterprise Corpora.” In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’06)*. ACM, Aug. 2006, pp. 43–50 (cit. on p. 38).

[Bar91] J. Barney. “Firm Resources and Sustained Competitive Advantage.” In: *Journal of Management* 17.1 (1991), pp. 99–120 (cit. on p. 32).

[BB14] P. Bell and B. Beer. *Introducing GitHub: A Non-Technical Guide*. O’Reilly Media, Inc, 2014 (cit. on p. 159).

[BBH+13] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner. “OpenTOSCA – A Runtime for TOSCA-based Cloud Applications.” In: *Service-Oriented Computing (ICSOC 2013)*. Springer, 2013, pp. 692–695 (cit. on pp. 62, 159).

[BBK+12] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and D. Schumm. “Vino4TOSCA: A Visual Notation for Application Topologies Based on TOSCA.” In: *On the Move to Meaningful Internet Systems: OTM 2012*. Springer, Sept. 2012, pp. 416–424 (cit. on pp. 60, 61, 102, 103).

[BBK+14] U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, and J. Wettinger. "Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA." In: *2014 IEEE International Conference on Cloud Engineering (IC2E 2014)*. IEEE, Apr. 2014, pp. 87–96 (cit. on pp. 61, 62, 195).

[BBK+16] U. Breitenbücher, T. Binz, O. Kopp, K. Képes, F. Leymann, and J. Wettinger. "Hybrid TOSCA Provisioning Plans: Integrating Declarative and Imperative Cloud Application Provisioning Technologies." In: *Cloud Computing and Services Science. CLOSER 2015*. Springer, May 2016, pp. 239–262 (cit. on pp. 62, 195).

[BBKL13] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann. "Automated Discovery and Maintenance of Enterprise Topology Graphs." In: *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications (SOCA 2013)*. IEEE, Dec. 2013, pp. 126–134 (cit. on p. 62).

[BBKL14a] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann. "TOSCA: Portable Automated Deployment and Management of Cloud Applications." In: *Advanced Web Services*. Springer New York, 2014, pp. 527–549 (cit. on p. 60).

[BBKL14b] U. Breitenbücher, T. Binz, O. Kopp, and F. Leymann. "Automating Cloud Application Management Using Management Idioms." In: *Proceedings of the Sixth International Conference on Pervasive Patterns and Applications (PATTERNS 2014)*. Xpert Publishing Services, May 2014, pp. 60–69 (cit. on p. 61).

[BBLS12] T. Binz, G. Breiter, F. Leymann, and T. Spatzier. "Portable Cloud Services Using TOSCA." In: *IEEE Internet Computing* 16.03 (May 2012), pp. 80–85 (cit. on p. 59).

[BCM+15] S. Bala, C. Cabanillas, J. Mendling, A. Rogge-Solti, and A. Polleres. "Mining Project-Oriented Business Processes." In: *Business Process Management: BPM 2015*. Springer, Aug. 2015, pp. 425–440 (cit. on p. 38).

[BDF+03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the Art of Virtualization." In: *SIGOPS Oper. Syst. Rev.* 37.5 (Oct. 2003), pp. 164–177 (cit. on p. 58).

[Bec09] S. Bechhofer. "OWL: Web ontology language." In: *Encyclopedia of Database Systems*. Springer, 2009, pp. 2008–2009 (cit. on p. 47).

[BFKR14] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg. "How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective." In: *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering* 8.1 (2014), pp. 37–44 (cit. on pp. 12, 20).

[BFV11] M. Brambilla, P. Fraternali, and C. Vaca. "A Notation for Supporting Social Business Process Modeling." In: *Business Process Model and Notation: BPMN 2011*. Springer, Nov. 2011 (cit. on p. 42).

[BFV12] M. Brambilla, P. Fraternali, and C. Vaca. "BPMN and Design Patterns for Engineering Social BPM Solutions." In: *Business Process Management Workshops: BPM 2011 International Workshops*. Springer, Aug. 2012 (cit. on p. 42).

[BFVB12] M. Brambilla, P. Fraternali, C. Vaca, and S. Butti. "Combining social web and BPM for improving enterprise performances: the BPM4People approach to social BPM." In: *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, Apr. 2012, pp. 223–226 (cit. on p. 42).

[Bha00] A. S. Bharadwaj. "A Resource-Based Perspective on Information Technology Capability and Firm Performance: An Empirical Investigation." In: *MIS Quarterly* 24.1 (Mar. 2000), pp. 169–196 (cit. on p. 32).

[Bin15] T. Binz. "Crawling von Enterprise Topologien zur automatisierten Migration von Anwendungen – eine Cloud-Perspektive." PhD thesis. University of Stuttgart, Apr. 2015 (cit. on pp. 31, 62, 115).

[BKH01] J. E. Barlett, J. W. Kotrlik, and C. C. Higgins. “Organizational research: Determining appropriate sample size in survey research.” In: *Information technology, learning, and performance journal* 19.1 (2001), p. 43 (cit. on p. 219).

[BKHM07] J. Bailey, E. Kandogan, E. Haber, and P. P. Maglio. “Activity-based management of IT service delivery.” In: *Proceedings of the 2007 symposium on Computer human interaction for the management of information technology (CHIMIT ’07)*. ACM, Apr. 2007 (cit. on pp. 12, 56).

[BKS11] C. Brecher, S. Kozielski, and L. Schapp. “Integrative Produktionstechnik für Hochlohnländer.” In: *Wertschöpfung und Beschäftigung in Deutschland*. Vol. 0. ACATECHDISK. Berlin, Heidelberg: Springer, 2011, pp. 47–70 (cit. on p. 43).

[BKZ10] A. Begel, Y. P. Khoo, and T. Zimmermann. “Codebook: discovering and exploiting relationships in software repositories.” In: *2010 ACM/IEEE 32nd International Conference on Software Engineering (ICSE ’10)*. IEEE, May 2010, pp. 125–134 (cit. on pp. 38, 196).

[BLMP09] A. Bucchiarone, A. L. Lafuente, A. Marconi, and M. Pistore. “A Formalisation of Adaptable Pervasive Flows.” In: *Web Services and Formal Methods*. Springer, Sept. 2009, pp. 61–75 (cit. on pp. 16, 45, 49, 52, 211, 212).

[BML+11] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao. “NIST Cloud Computing Reference Architecture.” In: *2011 IEEE World Congress on Services*. IEEE, July 2011, pp. 594–596 (cit. on p. 57).

[BMPR12] A. Bucchiarone, A. Marconi, M. Pistore, and H. Raik. “Dynamic Adaptation of Fragment-Based and Context-Aware Business Processes.” In: *2012 IEEE 19th International Conference on Web Services (ICWS 2012)*. IEEE, June 2012, pp. 33–41 (cit. on pp. 45, 49, 52).

[Bre16] U. Breitenbücher. “Eine musterbasierte Methode zur Automatisierung des Anwendungsmanagements.” PhD thesis. University of Stuttgart, Mar. 2016 (cit. on pp. 31, 63, 102, 103).

[Bun16] S. Bundesamt. *Statistisches Jahrbuch Deutschland und Internationales: 2016*. Artikelnummer: 1010110-16700-4. Oct. 2016 (cit. on p. 219).

[Bur04] W. Burr. *Innovationen in Organisationen*. Stuttgart: Kohlhammer, 2004 (cit. on pp. 20, 31, 32).

[Bur16] W. Burr. “Theoretische Grundlagen und Konzepte.” In: *Service Engineering bei technischen Dienstleistungen: Eine ökonomische Analyse der Modularisierung, Leistungstiefengestaltung und Systembündelung*. Springer, 2016. Chap. 2 (cit. on p. 31).

[BYV+09] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.” In: *Future Generation Computer Systems* 25.6 (June 2009), pp. 599–616 (cit. on p. 59).

[CGR+13] C. Cabanillas, J. M. García, M. Resinas, D. Ruiz, J. Mendling, and A. Ruiz-Cortés. “Priority-Based Human Resource Allocation in Business Processes.” In: *Service-Oriented Computing: ICSOC 2013*. Springer, Dec. 2013, pp. 374–388 (cit. on p. 40).

[Che16] H. Cheng. “Supporting Organizational Goals with Mobile Apps.” MA thesis. University of Stuttgart, Sept. 2016 (cit. on p. 229).

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3. The MIT Press, 2009 (cit. on p. 165).

[Coc77] W. G. Cochran. *Sampling techniques*. 3rd ed. John Wiley & Sons, Jan. 1977 (cit. on p. 219).

[CRdR15] C. Cabanillas, M. Resinas, A. del-Río-Ortega, and A. Ruiz-Cortés. “Specification and automated design-time analysis of the business process human resource perspective.” In: *Information Systems* 52 (Aug. 2015). Special Issue on Selected Papers from SISAP 2013, pp. 55–82 (cit. on p. 40).

[Cre13] J. W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 4th ed. SAGE Publications, Mar. 2013 (cit. on pp. 216, 219).

[CRMR15] C. Cabanillas, M. Resinas, J. Mendling, and A. Ruiz-Cortés. “Automated Team Selection and Compliance Checking in Business Processes.” In: *Proceedings of the 2015 International Conference on Software and System Process (ICSSP 2015)*. ACM, Aug. 2015, pp. 42–51 (cit. on pp. 40, 111).

[CRR12] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. “RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes.” In: *Business Process Management Workshops: BPM 2011 International Workshops*. Springer, Aug. 2012, pp. 50–61 (cit. on p. 40).

[CTD13] M. Z. Candra, H.-L. Truong, and S. Dustdar. “Provisioning quality-aware social compute units in the cloud.” In: *Service-Oriented Computing: ICSOC 2013*. Springer, Dec. 2013, pp. 313–327 (cit. on p. 58).

[Dav05] T. H. Davenport. *Thinking for a Living: How to Get Better Performances And Results from Knowledge Workers*. Harvard Business Press, Sept. 2005 (cit. on pp. 12, 16, 18, 20, 30).

[Dav10] T. H. Davenport. “Process Management for Knowledge Work.” In: *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Ed. by J. v. Brocke and M. Rosemann. Berlin, Heidelberg: Springer, 2010, pp. 17–35 (cit. on pp. 12, 17, 18, 31).

[Dav11] T. H. Davenport. “Rethinking knowledge work: A strategic approach.” In: *McKinsey Quarterly* (Feb. 2011) (cit. on p. 31).

[DB11] S. Dustdar and K. Bhattacharya. “The Social Compute Unit.” In: *IEEE Internet Computing* 15.3 (Apr. 2011), pp. 64–69 (cit. on p. 58).

[DD07] J. L. De la Vara González and J. S. Diaz. “Business process-driven requirements engineering: a goal-based approach.” In: *The 8th Workshop on Business Process Modeling, Development, and Support (BPMDS’07)*. Tapir Academic Press, June 2007, pp. 299–308 (cit. on p. 33).

[DD11] C. Dorn and S. Dustdar. “Supporting Dynamic, People-Driven Processes through Self-learning of Message Flows.” In: *Advanced Information Systems Engineering: CAiSE 2011*. Springer, June 2011, pp. 657–671 (cit. on pp. 38, 196).

[DDB98] T. H. Davenport, D. W. De Long, and M. C. Beers. “Successful Knowledge Management Projects.” In: *Sloan Management Review* 39.2 (1998), pp. 43–58 (cit. on pp. 32, 33).

[Dey01] A. K. Dey. “Understanding and Using Context.” In: *Personal and Ubiquitous Computing* 5.1 (Feb. 2001), pp. 4–7 (cit. on p. 44).

[DMR15] C. Di Ciccio, A. Marrella, and A. Russo. “Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches.” In: *Journal on Data Semantics* 4.1 (Mar. 2015), pp. 29–57 (cit. on pp. 12, 14–17, 30–33).

[DP98] T. H. Davenport and L. Prusak. *Working Knowledge: How Organizations Manage what They Know*. Harvard Business Press, 1998 (cit. on p. 32).

[DR09] P. Dadam and M. Reichert. “The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements.” In: *Computer Science - Research and Development* 23.2 (May 2009), pp. 81–97 (cit. on pp. 41, 110).

[Dre81] F. Dretske. *Knowledge and the Flow of Information*. MIT Press, 1981 (cit. on p. 32).

[dSdSPvS09] L. O. B. d. S. Santos, E. G. d. Silva, L. F. Pires, and M. v. Sinderen. “Towards a Goal-Based Service Framework for Dynamic Service Discovery and Composition.” In: *2009 Sixth International Conference on Information Technology: New Generations (ITNG ’09)*. IEEE, Apr. 2009, pp. 302–307 (cit. on pp. 33, 47, 212).

[DT12a] C. Dorn and R. N. Taylor. “Architecture-Driven Modeling of Adaptive Collaboration Structures in Large-Scale Social Web Applications.” In: *Web Information Systems Engineering: WISE 2012*. Springer, Nov. 2012, pp. 143–156 (cit. on p. 34).

[DT12b] C. Dorn and R. N. Taylor. “Co-adapting human collaborations and software architectures.” In: *2012 34th International Conference on Software Engineering (ICSE 2012)*. IEEE, June 2012, pp. 1277–1280 (cit. on p. 34).

[DT12c] S. Dustdar and H. L. Truong. “Virtualizing Software and Humans for Elastic Processes in Multiple Clouds– a Service Management Perspective.” In: *International Journal of Next-Generation Computing* 3.2 (July 2012), pp. 109–126 (cit. on p. 58).

[Dun66] O. D. Duncan. “Path analysis: Sociological examples.” In: *American Journal of Sociology* 72.1 (July 1966), pp. 1–16 (cit. on p. 217).

[Dus04] S. Dustdar. “Caramba—A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams.” In: *Distributed and Parallel Databases* 15.1 (Jan. 2004), pp. 45–66 (cit. on pp. 43, 109, 110, 112–114).

[Dus05] S. Dustdar. “Architecture and design of an internet-enabled integrated workflow and groupware system.” In: *Business Process Management Journal* 11.3 (2005), pp. 275–290 (cit. on pp. 43, 109, 110, 112–114).

[DVT05] M. Dumas, W. M. P. Van der Aalst, and A. H. M. Ter Hofstede. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, Oct. 2005 (cit. on pp. 11, 34).

[DWHB15] J. Deuse, K. Weisner, A. Hengstebeck, and F. Busch. “Gestaltung von Produktionssystemen im Kontext von Industrie 4.0.” In: *Zukunft der Arbeit in Industrie 4.0*. Springer, 2015, pp. 99–109 (cit. on p. 201).

[Ebe14] H. Eberle. “Prozessbausteine.” Dissertation. University of Stuttgart, Jan. 2014, p. 324 (cit. on pp. 16, 46).

[EHN94] K. Erol, J. Hendler, and D. S. Nau. “HTN planning: Complexity and expressivity.” In: *The Twelfth National Conference on Artificial Intelligence (AAAI’94)*. AAAI Press, July 1994, pp. 1123–1128 (cit. on pp. 49, 50).

[EHN95] K. Erol, J. A. Hendler, and D. S. Nau. *Semantics for hierarchical task-network planning*. Tech. rep. Institute for Systems Research Technical Reports, 1995 (cit. on p. 49).

[EL16] W. Engelbert and C. Löffler. *Strategien der Produktion*. Springer, 2016 (cit. on pp. 12, 20, 43, 200).

[EP98] H.-E. Eriksson and M. Penker. *Business Modeling With UML: Business Patterns at Work*. New York, NY, USA: John Wiley & Sons, 1998 (cit. on p. 32).

[Erl13] K. Erlach. *Value stream design: The Way Towards a Lean Factory*. Springer, 2013 (cit. on p. 234).

[Ero96] K. Erol. “Hierarchical Task Network Planning: Formalization, Analysis, And Implementation.” PhD thesis. University of Maryland, 1996 (cit. on p. 49).

[FBB+14] M. Falkenthal, J. Barzen, U. Breitenbücher, C. Fehling, and F. Leymann. “Efficient Pattern Application: Validating the Concept of Solution Implementations in Different Domains.” In: *International Journal on Advances in Software* 7.3&4 (Dec. 2014), pp. 710–726 (cit. on p. 59).

[FBV11] P. Fraternali, M. Brambilla, and C. Vaca. “A Model-driven Approach to Social BPM Applications.” In: *Social BPM* (2011) (cit. on p. 42).

[FFP12] J. M. Fernández-de-Alba, R. Fuentes-Fernández, and J. Pavón. “Dynamic Workflow Management for Context-Aware Systems.” In: *Ambient Intelligence - Software and Applications (ISAmI 2012)*. Springer, June 2012, pp. 181–188 (cit. on p. 45).

[FFST11] D. Fensel, F. M. Facca, E. Simperl, and I. Toma. “Web Service Modeling Ontology.” In: *Semantic Web Services*. Berlin, Heidelberg: Springer, 2011, pp. 107–129 (cit. on pp. 47, 212).

[FGP14] F. Folino, M. Guarascio, and L. Pontieri. “Mining Predictive Process Models out of Low-level Multidimensional Logs.” In: *Advanced Information Systems Engineering: CAiSE 2014*. Vol. 8484. Springer, June 2014, pp. 533–547 (cit. on p. 37).

[FGP15a] F. Folino, M. Guarascio, and L. Pontieri. “Mining Multi-variant Process Models from Low-Level Logs.” In: *Business Information Systems: BIS 2015*. Springer, June 2015, pp. 165–177 (cit. on p. 37).

[FGP15b] F. Folino, M. Guarascio, and L. Pontieri. “On the Discovery of Explainable and Accurate Behavioral Models for Complex Lowly-structured Business Processes.” In: *Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS 2015)*. SciTePress, Apr. 2015, pp. 206–217 (cit. on p. 37).

[Flä15] H. Flämig. “Autonome Fahrzeuge und autonomes Fahren im Bereich des Gütertransportes.” In: *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*. Ed. by M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner. Springer, 2015, pp. 377–398 (cit. on p. 238).

[FLR+14] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, 2014 (cit. on p. 59).

[FN71] R. E. Fikes and N. J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving." In: *Artificial intelligence* 2.3-4 (Sept. 1971), pp. 189–208 (cit. on p. 49).

[GA15] I. Georgievski and M. Aiello. "HTN planning: Overview, comparison, and beyond." In: *Artificial Intelligence* 222 (May 2015), pp. 124–156 (cit. on p. 49).

[GAH+15] S. Gómez Sáez, V. Andrikopoulos, M. Hahn, D. Karastoyanova, and A. Weiß. "Enabling Reusable and Adaptive Modeling, Provisioning & Execution of BPEL Processes." In: *2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA 2015)*. IEEE, Oct. 2015, pp. 51–58 (cit. on p. 49).

[Gal05] J. Galbreath. "Which resources matter the most to firm success? An exploratory study of resource-based theory." In: *Technovation* 25.9 (2005), pp. 979–987 (cit. on p. 32).

[GBMP13] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions." In: *Future Generation Computer Systems* 29.7 (Sept. 2013), pp. 1645–1660 (cit. on p. 44).

[Gla12] L. Glavan. "Understanding Process Performance Measurement Systems." In: *Business Systems Research* 2.2 (Sept. 2012), pp. 25–38 (cit. on p. 48).

[GMNS03] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. "Reasoning with Goal Models." In: *Conceptual Modeling: ER 2002*. Springer, Oct. 2003, pp. 167–181 (cit. on p. 33).

[GOR10] G. Grambow, R. Oberhauser, and M. Reichert. "Semantic Workflow Adaption in Support of Workflow Diversity." In: *4th International Conference on Advances in Semantic Processing (SEMAPRO'10)*. Xpert Publishing Services, Nov. 2010, pp. 158–165 (cit. on p. 51).

[GOR11] G. Grambow, R. Oberhauser, and M. Reichert. “Semantically-Driven Workflow Generation Using Declarative Modeling for Processes in Software Engineering.” In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops (EDOCW 2011*. IEEE, Aug. 2011, pp. 158–165 (cit. on pp. 12, 18, 51).

[Gra96] R. M. Grant. “Prospering in Dynamically-Competitive Environments: Organizational Capability as Knowledge Integration.” In: *Organization Science* 7.4 (Aug. 1996), pp. 375–387 (cit. on p. 32).

[GRG+04] J. M. Gomez, M. Rico, F. García-Sanchez, I. Toma, and S.-K. Han. “Godo: Goal oriented discovery for semantic web services.” In: *WSMO Implementation Workshop 2004 (WIW 2004)*. Vol. 113. CEUR-WS, Sept. 2004, pp. 1–11 (cit. on pp. 47, 212).

[Gro16] T. O. Group. *ArchiMate 3. 0 Specification*. Open Group Standard. 2016. URL: `http://pubs.opengroup.org/architecture/archimate3-doc/` (cit. on pp. 34, 40).

[GSBC00] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki. “Managing escalation of collaboration processes in crisis mitigation situations.” In: *Proceedings of 16th International Conference on Data Engineerin (ICDE 2000)*. IEEE, Mar. 2000, pp. 45–56 (cit. on pp. 49, 211, 212).

[Gua98] N. Guarino. “Formal Ontology and Information Systems.” In: *Proceedings of FOIS’98*. IOS Press, June 1998, pp. 3–15 (cit. on p. 47).

[GW96] K. Golden and D. Weld. “Representing sensing actions: The middle ground revisited.” In: *Principles of Knowledge Representation and Reasoning: KR ’96*. Morgan Kaufmann Publishers, Nov. 1996, pp. 174–185 (cit. on p. 49).

[Hab08] I. Habib. “Virtualization with KVM.” In: *Linux J.* 2008.166 (Feb. 2008) (cit. on p. 58).

[HBBL14] P. Hirmer, U. Breitenbücher, T. Binz, and F. Leymann. “Automatic Topology Completion of TOSCA-based Cloud Applications.” In: *CloudCycle14 Workshops*. Gesellschaft für Informatik e.V. (GI), Sept. 2014, pp. 247–258 (cit. on p. 61).

[HBJ+14] J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos. "Strategic business modeling: representation and reasoning." In: *Software & Systems Modeling* 13.3 (July 2014), pp. 1015–1041 (cit. on p. 33).

[HBR08] A. Hallerbach, T. Bauer, and M. Reichert. "Context-based Configuration of Process Variants." In: *3rd International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008)*. INSTICC Press, June 2008, pp. 31–40 (cit. on p. 48).

[HBR10a] A. Hallerbach, T. Bauer, and M. Reichert. "Capturing Variability in Business Process Models: The Provop Approach." In: *Journal of Software Maintenance and Evolution: Research and Practice* 22.6-7 (Nov. 2010), pp. 519–546 (cit. on pp. 47, 48).

[HBR10b] A. Hallerbach, T. Bauer, and M. Reichert. "Configuration and Management of Process Variants." In: *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer, 2010, pp. 237–255 (cit. on p. 48).

[HCMP16] G. Havur, C. Cabanillas, J. Mendling, and A. Polleres. "Resource Allocation with Dependencies in Business Process Management Systems." In: *Business Process Management Forum: BPM Forum 2016*. Springer, Sept. 2016, pp. 3–19 (cit. on pp. 49, 212).

[HK11] C. Herrmann and M. Kurz. "Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems." In: *S-BPM ONE – Learning by Doing – Doing by Learning: S-BPM ONE 2011*. Springer, Sept. 2011, pp. 80–97 (cit. on pp. 33, 56, 114).

[HKZ15] B. Heinrich, M. Klier, and S. Zimmermann. "Automated planning of process models: Design of a novel approach to construct exclusive choices." In: *Decision Support Systems* 78 (Oct. 2015), pp. 1–14 (cit. on p. 49).

[HS15] B. Heinrich and D. Schön. "Automated Planning of context-aware Process Models." In: *European Conference on Information Systems (ECIS 2015)*. Paper 75. AIS eLibrary, May 2015 (cit. on pp. 49, 212).

[HTZD08] T. Holmes, H. Tran, U. Zdun, and S. Dustdar. “Modeling Human Aspects of Business Processes – A View-Based, Model-Driven Approach.” In: *Model Driven Architecture –– Foundations and Applications: ECMDA-FA 2008*. Springer, June 2008, pp. 246–261 (cit. on p. 42).

[HW04] G. Hohpe and B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004 (cit. on pp. 59, 227, 233).

[HZJ08] G. Q. Huang, Y. Zhang, and P. Jiang. “RFID-based wireless manufacturing for real-time management of job shop WIP inventories.” In: *The International Journal of Advanced Manufacturing Technology* 36.7 (Mar. 2008), pp. 752–764 (cit. on p. 201).

[IA10] C. Ibsen and J. Anstey. *Camel in Action*. Manning Publications Co., 2010 (cit. on p. 233).

[IMF+10] M. Igler, P. Moura, M. Faerber, M. Zeising, and S. Jablonski. “Modeling and planning collaboration using organizational constraints.” In: *6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010)*. IEEE, Oct. 2010, pp. 1–10 (cit. on pp. 52, 53).

[IMZJ10] M. Igler, P. Moura, M. Zeising, and S. Jablonski. “ESProNa: Constraint-Based Declarative Business Process Modeling.” In: *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW 2010)*. IEEE, Nov. 2010, pp. 91–98 (cit. on pp. 52, 53).

[Isr92] G. D. Israel. *Determining Sample Size*. Fact Sheet PEOD-6. University of Florida Cooperative Extension Service, Institute of Food and Agriculture Sciences, EDIS, 1992 (cit. on p. 219).

[JLvB+04] H. Jonkers, M. Lankhorst, R. van Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. van der Torre. “Concepts For Modeling Enterprise Architectures.” In: *International Journal of Cooperative Information Systems* 13.03 (Sept. 2004), pp. 257–287 (cit. on p. 34).

[Kal16] A. Kalidoss. “Intention-oriented Organizational Modeling – A Top-down Approach.” MA thesis. University of Stuttgart, Aug. 2016 (cit. on p. 229).

[Kar16] D. Kar. “Goal-driven Context-sensitive Production Processes: A Case Study using BPMN.” MA thesis. University of Stuttgart, Mar. 2016 (cit. on p. 234).

[KBBL12] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann. “BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications.” In: *Business Process Model and Notation: BPMN 2012*. Springer, Sept. 2012, pp. 38–52 (cit. on p. 62).

[KBBL13] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann. “Winery – Modeling Tool for TOSCA-based Cloud Applications.” In: *Service-Oriented Computing: ICSOC 2013*. Springer, Dec. 2013, pp. 700–704 (cit. on p. 60).

[KBG+16] K. Képes, U. Breitenbücher, S. Gómez Sáez, J. Guth, F. Leymann, and M. Wieland. “Situation-Aware Execution and Dynamic Adaptation of Traditional Workflow Models.” In: *Service-Oriented and Cloud Computing: ESOCC 2016*. Springer, Sept. 2016, pp. 69–83 (cit. on pp. 46, 211).

[KGT06] A. Knopfel, B. Grone, and P. Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems*. John Wiley & Sons, Ltd., 2006 (cit. on p. 233).

[KK97] P. Kueng and P. Kawalek. “Goal-based business process models: creation and evaluation.” In: *Business Process Management Journal* 3.1 (1997), pp. 17–38 (cit. on p. 33).

[KLA11] E. Kaldeli, A. Lazovik, and M. Aiello. “Continual Planning with Sensing for Web Service Composition.” In: *The Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI’11)*. AAAI Press, Aug. 2011, pp. 1198–1203 (cit. on pp. 50, 212).

[KLA16] E. Kaldeli, A. Lazovik, and M. Aiello. “Domain-independent planning for services in uncertain and dynamic environments.” In: *Artificial Intelligence* 236 (July 2016), pp. 30–64 (cit. on pp. 50, 212).

[KT16] A. Khan and K. Turowski. “A Survey of Current Challenges in Manufacturing Industry and Preparation for Industry 4.0.” In: *Proceedings of the First International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’16)*. Springer, 2016, pp. 15–26 (cit. on p. 229).

[KW03] F. Keller and S. Wendt. “FMC: an approach towards architecture-centric system development.” In: *10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2003. Proceedings (ECBS 2003)*. IEEE, Apr. 2003, pp. 173–182 (cit. on p. 233).

[KW10] A. Kumar and J. Wang. “A Framework for Designing Resource-Driven Workflows.” In: *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer, 2010, pp. 419–440 (cit. on pp. 56, 113).

[Lan+13] M. Landherr et al. “Fabriklebenszyklusmanagement.” In: *Digitale Produktion*. Springer, 2013, pp. 163–195 (cit. on p. 200).

[LAP06] A. Lazovik, M. Aiello, and M. Papazoglou. “Planning and monitoring the execution of web service requests.” In: *International Journal on Digital Libraries* 6.3 (2006), pp. 235–246 (cit. on p. 50).

[LCG+09] R. Lange, N. Cipriani, L. Geiger, M. Grossmann, H. Weinschrott, A. Brodt, M. Wieland, S. Rizou, and K. Rothermel. “Making the World Wide Space happen: New challenges for the Nexus context platform.” In: *2009 IEEE International Conference on Pervasive Computing and Communications (PERCOM 2009)*. IEEE, Mar. 2009, pp. 1–4 (cit. on pp. 45, 207, 233).

[LCW08] D. Lucke, C. Constantinescu, and E. Westkämper. “Smart factory – a step towards the next generation of manufacturing.” In: *Manufacturing Systems and Technologies for the New Frontier (CIRP CMS 2008)*. Springer, May 2008, pp. 115–118 (cit. on p. 44).

[Ley09] F. Leymann. “Cloud Computing: The Next Revolution in IT.” In: *Photogrammetric Week ‘09*. Wichmann Verlag, 2009, pp. 3–12 (cit. on pp. 57, 58).

[Ley12] F. Leymann. “Linked Compute Units and Linked Experiments: Using Topology and Orchestration Technology for Flexible Support of Scientific Applications.” In: *Software Service and Application Engineering*. Springer, 2012, pp. 71–80 (cit. on p. 33).

[LFK+14] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann. “Industry 4.0.” In: *Business & Information Systems Engineering* 6.4 (Aug. 2014), 239–242 (cit. on pp. 12, 20, 43, 47).

[LFWW16] F. Leymann, C. Fehling, S. Wagner, and J. Wettinger. “Native Cloud Applications: Why Virtual Machines, Images and Containers Miss the Point!” In: *8th International Conference on Cloud Computing and Services Science (CLOSER 2016)*. SciTePress, Apr. 2016, 7–15 (cit. on pp. 58, 59).

[LKTD12] V. Liptchinsky, R. Khazankin, H.-L. Truong, and S. Dustdar. “A Novel Approach to Modeling Context-Aware and Social Collaboration Processes.” In: *Advanced Information Systems Engineering: CAiSE 2012*. Springer, June 2012, pp. 565–580 (cit. on pp. 55, 112).

[LLT09] T. Lappas, K. Liu, and E. Terzi. “Finding a Team of Experts in Social Networks.” In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’09)*. ACM, June 2009, pp. 467–476 (cit. on p. 38).

[LM12] J. Loeliger and M. McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development*. "OReilly Media, Inc.", 2012 (cit. on p. 38).

[LR00] F. Leymann and D. Roller. *Production Work Flow: Concepts and Techniques*. Prentice Hall PTR, 2000 (cit. on pp. 11, 15, 24, 30–33, 40, 70, 203).

[LRDR05] L. Ly, S. Rinderle, P. Dadam, and M. Reichert. “Mining Staff Assignment Rules from Event-Based Data.” In: *Business Process Management Workshops: BPM 2005 International Workshops*. Springer, Nov. 2005, pp. 177–190 (cit. on p. 37).

[LRS02] F. Leymann, D. Roller, and M.-T. Schmidt. “Web services and business process management.” In: *IBM Systems Journal* 41.2 (Apr. 2002), pp. 198–211 (cit. on p. 36).

[Mar+05] D. Martin et al. “Bringing Semantics to Web Services: The OWL-S Approach.” In: *Semantic Web Services and Web Process Composition: SWSWPC 2004*. Springer, June 2005, pp. 26–42 (cit. on pp. 50, 212).

[Mar91] J. G. March. “Exploration and Exploitation in Organizational Learning.” In: *Organization Science* 2.1 (1991), pp. 71–87 (cit. on p. 16).

[MC10] I. Mirbel and P. Crescenzo. “From End-User’s Requirements to Web Services Retrieval: A Semantic and Intention-Driven Approach.” In: *Exploring Services Science: IESS 2010*. Springer, Feb. 2010, pp. 30–44 (cit. on pp. 47, 212).

[MCF05] T. P. Moran, A. Cozzi, and S. P. Farrell. “Unified Activity Management: Supporting People in E-business.” In: *Communications of the ACM* 48.12 (Dec. 2005), pp. 67–70 (cit. on pp. 12, 56).

[MG11] P. M. Mell and T. Grance. *SP 800-145. The NIST Definition of Cloud Computing*. Tech. rep. National Institute of Standards & Technology, 2011 (cit. on pp. 57, 58).

[MGKR15] N. Mundbrod, G. Grambow, J. Kolb, and M. Reichert. “Context-Aware Process Injection.” In: *On the Move to Meaningful Internet Systems: OTM 2015 Conferences*. Springer, Oct. 2015, pp. 127–145 (cit. on pp. 46, 211).

[MGM+06] P. Moody, D. Gruen, M. Muller, J. Tang, and T. Moran. “Business activity patterns: A new model for collaborative business applications.” In: *IBM Systems Journal* 45.4 (Apr. 2006), pp. 683–694 (cit. on pp. 12, 18, 56).

[MLB+11] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi. “Cloud computing – The business perspective.” In: *Decision Support Systems* 51.1 (Apr. 2011), pp. 176–189 (cit. on p. 57).

[Mob02] R. K. Mobley. *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, 2002 (cit. on p. 200).

[Moh73] L. B. Mohr. “The Concept of Organizational Goal.” In: *The American Political Science Review* 67.2 (1973), pp. 470–481 (cit. on p. 33).

[Moo09] D. Moody. “The “Physics” of “Notations”: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering.” In: *IEEE Transactions on Software Engineering* 35.6 (Dec. 2009), pp. 756–779 (cit. on p. 106).

[MR14] E. Muüller and R. Riedel. “Humanzentrierte Entscheidungsunterstützung in intelligent vernetzten Produktionssysteme.” In: *Industrie 4.0 Wie intelligente Vernetzung und kognitive Systeme unsere Arbeit verändern* (Sept. 2014), pp. 211–238 (cit. on p. 201).

[MRM12] A. Marrella, A. Russo, and M. Mecella. “Planlets: Automatically Recovering Dynamic Processes in YAWL.” In: *On the Move to Meaningful Internet Systems: OTM 2012*. Springer, Sept. 2012, pp. 268–286 (cit. on p. 52).

[MS13] H. R. Motahari-Nezhad and K. D. Swenson. “Adaptive Case Management: Overview and Research Challenges.” In: *2013 IEEE 15th Conference on Business Informatics (CBI 2013)*. IEEE, July 2013, pp. 264–269 (cit. on pp. 56, 114).

[MSB+13] H. Motahari-Nezhad, S. Spence, C. Bartolini, S. Graupner, C. Bess, M. Hickey, P. Joshi, R. Mirizzi, K. Ozonat, and M. Rahmouni. “Casebook: A Cloud-Based System of Engagement for Case Management.” In: *IEEE Internet Computing* 17.5 (May 2013), pp. 30–38 (cit. on pp. 56, 114).

[MST09] S. Meyer, H. Simon, and M. Tilebein. “Applying Agent-Based Modeling to Integrate Bounded Rationality in Organizational Management Research.” In: *2009 42nd Hawaii International Conference on System Sciences (HICSS’09)*. IEEE, Jan. 2009, pp. 1–10 (cit. on p. 31).

[MWM+12] T. Matthews, S. Whittaker, T. P. Moran, S. Y. Helsley, and T. K. Judge. “Productive Interrelationships Between Collaborative Groups Ease the Challenges of Dynamic and Multi-Teaming.” In: *Computer Supported Cooperative Work* 21.4 (Dec. 2012), 371–396 (cit. on p. 33).

[MWMY11] T. Matthews, S. Whittaker, T. Moran, and S. Yuen. “Collaboration Personas: A New Approach to Designing Workplace Collaboration Tools.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI ’11)*. ACM, May 2011, pp. 2247–2256 (cit. on pp. 31, 33).

[NC03] A. Nigam and N. S. Caswell. “Business artifacts: An approach to operational specification.” In: *IBM Systems Journal* 42.3 (2003), pp. 428–445 (cit. on p. 54).

[NEK+05] S. Nurcan, A. Etien, R. Kaabi, I. Zoukar, and C. Rolland. “A Strategy driven business process modelling approach.” In: *Business Process Management Journal* 11.6 (2005), 628–649 (cit. on pp. 52–54, 238).

[Non94] I. Nonaka. “A Dynamic Theory of Organizational Knowledge Creation.” In: *Organization Science* 5.1 (Feb. 1994), pp. 14–37 (cit. on pp. 32, 33).

[Nur08] S. Nurcan. “A survey on the flexibility requirements related to business processes and modeling artifacts.” In: *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. Jan. 2008, pp. 378–388 (cit. on pp. 11, 41, 54).

[OAÇ06] S. Orhan, N. Aktürk, and V. Çelik. “Vibration monitoring for defect diagnosis of rolling element bearings as a predictive maintenance tool: Comprehensive case studies.” In: *NDT & E International* 39.4 (June 2006), pp. 293–298 (cit. on p. 200).

[OAS07a] OASIS. *Web Services Resource 1.2*. OASIS Standard. 2007. URL: `http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf` (cit. on p. 36).

[OAS07b] OASIS. *Web Services Business Process Execution Language Version 2.0*. OASIS Standard. 2007. URL: `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html` (cit. on pp. 15, 41, 86, 198).

[OAS10a] OASIS. *Web Services Human Task (WS-HumanTask) Specification Version 1.1*. OASIS Standard. 2010. URL: `http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cs-01.pdf` (cit. on pp. 42, 87, 110, 112, 113).

[OAS10b] OASIS. *WS-BPEL Extension for People (BPEL4People) Specification Version 1.1*. OASIS Standard. 2010. URL: `http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html` (cit. on p. 42).

[OAS13a] OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0*. OASIS Committee Note Draft 01. 2013. URL: `http://docs.oasis-open.org/tosca/tosca-primer/v1.0/cnd01/tosca-primer-v1.0-cnd01.pdf` (cit. on p. 62).

[OAS13b] OASIS. *Topology and Orchestration Specification for Cloud Applications Version 1.0*. OASIS Standard. 2013. URL: `http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html` (cit. on pp. 59, 62).

[Obj11] Object Management Group (OMG). *Business Process Model and Notation Version (BPMN) Version 2.0*. OMG document number: formal/2011-01-03. 2011. URL: `http://www.omg.org/spec/BPMN/2.0/` (cit. on pp. 15, 41, 71, 119, 198).

[Obj16] Object Management Group (OMG). *Case Management Model and Notation (CMMN) Version 1.1*. OMG document number: formal/2016-12-01. 2016. URL: `http://www.omg.org/spec/CMMN/1.1/` (cit. on pp. 12, 55).

[OLe00] D. E. O'Leary. *Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce, and Risk*. Cambridge University Press, 2000 (cit. on p. 227).

[Oxf17] Oxford University Press. Online Dictionary. 2017. URL: `https://en.oxforddictionaries.com/definition/us/resource` (cit. on p. 12).

[PBMW99] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999–66. Stanford InfoLab, Nov. 1999 (cit. on p. 38).

[PCF08] C. M. Pilato, B. Collins-Sussman, and B. Fitzpatrick. *Version Control with Subversion*. "OReilly Media, Inc.", 2008 (cit. on p. 38).

[PSSvdA07] M. Pesic, M. H. Schonenberg, N. Sidorova, and W. M. P. van der Aalst. "Constraint-Based Workflow Models: Change Made Easy." In: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*. Ed. by R. Meersman and Z. Tari. Springer, Nov. 2007, pp. 77–94 (cit. on p. 51).

[Rad12] T. Rademakers. *Activiti in Action: Executable Business Processes in BPMN 2.0*. Manning Publications Co., 2012 (cit. on p. 232).

[RLG+15] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch. *Industry 4.0: The future of productivity and growth in manufacturing industries*. Tech. rep. Boston Consulting Group, 2015, p. 14 (cit. on p. 43).

[Rot16] A. Roth. "Industrie 4.0 – Hype oder Revolution?" In: *Einführung und Umsetzung von Industrie 4.0: Grundlagen, Vorgehensmodell und Use Cases aus der Praxis*. Springer, 2016, pp. 1–15 (cit. on p. 200).

[RtHEvdA04] N. Russell, A. H. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. *Workflow resource patterns*. Tech. rep. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004 (cit. on pp. 31, 111).

[RvdAtHE05] N. Russell, W. van der Aalst, A. H. ter Hofstede, and D. Edmond. "Workflow resource patterns: Identification, representation and tool support." In: *Advanced Information Systems Engineering*. Springer. 2005, pp. 216–232 (cit. on p. 40).

[RW12] M. Reichert and B. Weber. "Constraint-Based Process Models." In: *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer, 2012. Chap. 12, pp. 341–374 (cit. on p. 51).

[RZ94] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, 1994 (cit. on p. 33).

[SBBL14] C. T. Sungur, T. Binz, U. Breitenbücher, and F. Leymann. "Informal Process Essentials." In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference (EDOC 2014)*. IEEE, Sept. 2014, pp. 200–209 (cit. on p. 26).

[SBK+16] C. T. Sungur, U. Breitenbücher, O. Kopp, F. Leymann, M. Song, A. Weiß, C. Mayr-Dorn, and S. Dustdar. "Identifying Relevant Resources and Relevant Capabilities of Collaborations – A Case Study." In: *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW 2016)*. IEEE, Nov. 2016, pp. 1–4 (cit. on p. 27).

[SBK+17] C. T. Sungur, U. Breitenbücher, O. Kopp, F. Leymann, and A. Weiß. "Identifying Relevant Resources and Relevant Capabilities of Informal Processes." In: *19th International Conference on Enterprise Information Systems (ICEIS 2017)*. SciTePress, Apr. 2017, pp. 295–307 (cit. on p. 27).

[SBLW15a] C. T. Sungur, U. Breitenbücher, F. Leymann, and M. Wieland. "Context-sensitive Adaptive Production Processes." In: *48th CIRP Conference on Manufacturing Systems (CIRP CMS 2015)*. Vol. 41. Elsevier, June 2015, pp. 147–152 (cit. on p. 27).

[SBLW15b] C. T. Sungur, U. Breitenbücher, F. Leymann, and J. Wettinger. "Executing Informal Processes." In: *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS '15)*. ACM, Dec. 2015, 54:1–54:10 (cit. on p. 27).

[SCD+16] S. Schönig, C. Cabanillas, C. Di Ciccio, S. Jablonski, and J. Mendling. "Mining team compositions for collaborative work in business processes." In: *Software & Systems Modeling* (2016), pp. 1–19 (cit. on p. 37).

[Sch09] D. Schall. "Human Interactions in Mixed Systems - Architecture, Protocols, and Algorithms." PhD thesis. TU Wien, 2009 (cit. on p. 38).

[Sch12] D. Schall. "Expertise ranking using activity and contextual link measures." In: *Data & Knowledge Engineering* 71.1 (Jan. 2012), pp. 92–113 (cit. on pp. 38, 196).

[SDB10] D. Schall, S. Dustdar, and M. B. Blake. "Programming Human and Software-Based Web Services." In: *Computer* 43.7 (July 2010), pp. 82–85 (cit. on p. 43).

[SDDL15] C. T. Sungur, C. Dorn, S. Dustdar, and F. Leymann. "Transforming Collaboration Structures into Deployable Informal Processes." In: *Engineering the Web in the Big Data Era: ICWE 2015*. Springer, June 2015, pp. 231–250 (cit. on p. 27).

[SDTD09] D. Schall, C. Dorn, H.-L. Truong, and S. Dustdar. "On Supporting the Design of Human-Provided Services in SOA." In: *Service-Oriented Computing: ICSOC 2008 Workshops*. Springer, Dec. 2009, pp. 91–102 (cit. on pp. 42, 43).

[SF09] K. Swenson and J. Farris. "Human-Centered Business Process Management." In: *Fujitsu scientific and technical journal* 45.2 (Apr. 2009), pp. 160–170 (cit. on p. 30).

[SGDD07] D. Schall, R. Gombotz, C. Dorn, and S. Dustdar. "Human Interactions in Dynamic Environments through Mobile Web Services." In: *IEEE International Conference on Web Services (ICWS 2007)*. IEEE, July 2007, pp. 912–919 (cit. on p. 43).

[SGM04] R. Sebastiani, P. Giorgini, and J. Mylopoulos. “Simple and minimum-cost satisfiability for goal models.” In: *Advanced Information Systems Engineering*. Springer. 2004, pp. 20–35 (cit. on p. 33).

[SGM12] R. Sellami, W. Gaaloul, and S. Moalla. “An Ontology for Workflow Organizational Model Mining.” In: *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2012)*. IEEE, June 2012, pp. 199–204 (cit. on p. 37).

[SH03] J. Spillan and M. Hough. “Crisis Planning in Small Businesses:: Importance, Impetus and Indifference.” In: *European Management Journal* 21.3 (June 2003), pp. 398–407 (cit. on p. 20).

[Sha08] M. Sharma. *Elgg social networking*. Packt Publishing Ltd, 2008 (cit. on pp. 159, 227).

[Sie15] S. C. P. Sierra. “Investigating Informal Processes.” MA thesis. Tilburg University, University of Stuttgart, and University of Crete, June 2015 (cit. on p. 66).

[Sil11] B. Silver. *BPMN Method and Style, 2nd Edition, with BPMN Implementer’s Guide: A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0*. Cody-Cassidy Press, 2011 (cit. on p. 15).

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, Inc, 2012 (cit. on pp. 77, 78).

[SK10] M. Sonntag and D. Karastoyanova. “Next Generation Interactive Scientific Experimenting based on the Workflow Technology.” In: *From Proceeding (696) IASTED Technology Conferences*. 2010 (cit. on pp. 15, 41, 110).

[SKL14] C. T. Sungur, O. Kopp, and F. Leymann. “Supporting Informal Processes.” In: *The 6th Central European Workshop on Services and their Composition (ZEUS 2014)*. Vol. 1140. CEUR-WS, Feb. 2014, pp. 49–56 (cit. on p. 26).

[SL13] S. Song and S.-W. Lee. “A goal-driven approach for adaptive service composition using planning.” In: *Mathematical and Computer Modelling* 58.1–2 (July 2013), pp. 261–273 (cit. on pp. 50, 212).

[SLM+10] D. Schumm, F. Leymann, Z. Ma, T. Scheibler, and S. Strauch. "Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls." In: *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2010)*. Universitätsverlag Göttingen, Feb. 2010 (cit. on p. 16).

[SM96] D. Sloane and S. P. Morgan. "An introduction to categorical data analysis." In: *Annual review of sociology* 22.1 (Aug. 1996), pp. 351–375 (cit. on p. 218).

[Son16a] M. Song. "Recognition of Resource Patterns in Human-centric Processes: A Case Study." MA thesis. University of Stuttgart, Apr. 2016 (cit. on p. 229).

[Son16b] M. Sonntag. "Model-as-you-go - ein Ansatz zur flexiblen Entwicklung von wissenschaftlichen Workflows." PhD thesis. University of Stuttgart, 2016 (cit. on pp. 41, 110).

[SPW+04] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. "HTN planning for Web Service composition using SHOP2." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 1.4 (Oct. 2004), pp. 377–396 (cit. on pp. 50, 212).

[SSOK13] C. T. Sungur, P. Spiess, N. Oertel, and O. Kopp. "Extending BPMN for Wireless Sensor Networks." In: *2013 IEEE 15th Conference on Business Informatics (CBI 2013)*. IEEE, July 2013, pp. 109–116 (cit. on p. 45).

[STD08] D. Schall, H.-L. Truong, and S. Dustdar. "Unifying Human and Software Services in Web-Scale Collaborations." In: *IEEE Internet Computing* 12.3 (May 2008), pp. 62–68 (cit. on p. 43).

[STD11] D. Schall, H. Truong, and S. Dustdar. "The Human-Provided Services Framework." In: *Socially Enhanced Services Computing: Modern Models and Algorithms for Distributed Systems*. Springer, May 2011, pp. 1–15 (cit. on p. 43).

[STSJ08] P. Salhofer, G. Tretter, B. Stadlhofer, and F. Joanneum. "Goal-oriented Service Selection." In: *Proceedings of the 2nd international conference on Theory and practice of electronic governance (ICEGOV '08)*. ACM, Dec. 2008, pp. 60–66 (cit. on pp. 47, 212).

[SvdA08] M. Song and W. van der Aalst. "Towards comprehensive support for organizational mining." In: *Decision Support Systems* 46.1 (Dec. 2008), pp. 300–317 (cit. on p. 37).

[Tur14] J. Turnbull. *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2014 (cit. on pp. 58, 159).

[VCN+01] A. Vasconcelos, A. Caetano, J. Neves, P. Sinogas, R. Mendes, and J. Tribolet. "A framework for modeling strategy, business processes and information systems." In: *Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2001)*. IEEE, Sept. 2001, pp. 69–80 (cit. on p. 33).

[vdAal11] W. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011 (cit. on p. 37).

[vdAal16] W. van der Aalst. *Process Mining*. Springer, 2016 (cit. on pp. 34, 37).

[vdAal97] W. van der Aalst. *Exploring the process dimension of workflow management*. Coputing science reports; Vol. 9713. Eindhoven: Technische Universiteit Eindhoven, 1997 (cit. on p. 37).

[vdAPS09] W. van der Aalst, M. Pesic, and H. Schonenberg. "Declarative workflows: Balancing between flexibility and support." In: *Computer Science - Research and Development* 23.2 (May 2009), pp. 99–113 (cit. on pp. 51, 52).

[vdAS04] W. van der Aalst and M. Song. "Mining Social Networks: Uncovering Interaction Patterns in Business Processes." In: *Business Process Management: BPM 2004*. Springer, June 2004, pp. 244–260 (cit. on p. 37).

[vdAtH05] W. van der Aalst and A. ter Hofstede. "YAWL: yet another workflow language." In: *Information Systems* 30.4 (June 2005), pp. 245–275 (cit. on pp. 15, 41).

[vdAWG05] W. van der Aalst, M. Weske, and D. Grünbauer. "Case handling: a new paradigm for business process support." In: *Data & Knowledge Engineering* 53.2 (May 2005), pp. 129–162 (cit. on pp. 55, 114).

[vdAWM04] W. van der Aalst, T. Weijters, and L. Maruster. “Workflow mining: discovering process models from event logs.” In: *IEEE Transactions on Knowledge and Data Engineering* 16.9 (Sept. 2004), pp. 1128–1142 (cit. on p. 37).

[Ver08] K. Vergidis. “Business process optimisation using an evolutionary multi-objective framework.” PhD thesis. Cranfield University, 2008 (cit. on pp. 48, 212).

[vGG11] J. C. van Grondelle and M. Gülpers. “Specifying flexible business processes using pre and post conditions.” In: *The Practice of Enterprise Modeling: PoEM 2011*. Springer, Nov. 2011, pp. 38–51 (cit. on pp. 51, 52, 113).

[VKL13] K. Vukojevic-Haupt, D. Karastoyanova, and F. Leymann. “On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows (SOCA 2014).” In: *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*. IEEE, Dec. 2013, pp. 91–98 (cit. on p. 58).

[VRS05] W. Van Der Aalst, H. A. Reijers, and M. Song. “Discovering social networks from event logs.” In: *Computer Supported Cooperative Work* 14.6 (Dec. 2005), pp. 549–593 (cit. on p. 37).

[WAHK15] A. Weiß, V. Andrikopoulos, M. Hahn, and D. Karastoyanova. “Rewinding and Repeating Scientific Choreographies.” In: *On the Move to Meaningful Internet Systems: OTM 2015 Conferences*. Springer, Oct. 2015, pp. 337–347 (cit. on p. 58).

[Wai+13] T. Waizenegger et al. “Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing.” In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. Springer, Sept. 2013, pp. 360–376 (cit. on p. 238).

[Wal14] P. Walgenbach. “Organisation und Innovation.” In: *Innovationen: Theorien, Konzepte und Methoden der Innovationsforschung*. Ed. by W. Burr. Stuttgart: Kohlhammer, Sept. 2014, pp. 92–116 (cit. on p. 16).

[Wal99] B. Walters. “VMware Virtual Platform.” In: *Linux J.* 1999.63es (July 1999) (cit. on p. 58).

[WASL13] J. Wettinger, V. Andrikopoulos, S. Strauch, and F. Leymann. “Enabling Dynamic Deployment of Cloud Applications Using a Modular and Extensible PaaS Environment.” In: *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD 2013)*. IEEE, June 2013, pp. 478–485 (cit. on p. 59).

[WCL+05] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, 2005 (cit. on pp. 36, 43, 238).

[Wed16] B. Weder. *Einbindung von Informations-Ressourcen in informelle Prozesse*. Bachelor Thesis. May 2016 (cit. on p. 228).

[Wel94] D. S. Weld. “An introduction to least commitment planning.” In: *AI magazine* 15.4 (1994), p. 27 (cit. on p. 49).

[Wer84] B. Wernerfelt. “A resource-based view of the firm.” In: *Strategic Management Journal* 5.2 (Apr. 1984), pp. 171–180 (cit. on p. 32).

[Wes10] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2010 (cit. on pp. 11, 15, 34, 35, 53, 55).

[Wes13] E. Westkämper. “Struktureller Wandel durch Megatrends.” In: *Digitale Produktion*. Ed. by E. Westkämper, D. Spath, C. Constantinescu, and J. Lentes. Berlin, Heidelberg: Springer, 2013, pp. 7–9 (cit. on p. 12).

[WHR09] H. Wolf, K. Herrmann, and K. Rothermel. “Modeling Dynamic Context Awareness for Situated Workflows.” In: *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*. Springer, Nov. 2009, pp. 98–107 (cit. on pp. 45, 211).

[Wib13] M. Wibig. “Dynamic Programming and Genetic Algorithm for Business Processes Optimisation.” In: *International Journal of Intelligent Systems and Applications* 5.1 (Dec. 2013), pp. 44–51 (cit. on pp. 48, 212).

[Wie13] M. Wieland. “Methoden zur Modellierung und Ausführung kontextbezogener Workflows in Produktionsumgebungen.” Dissertation. University of Stuttgart, 2013 (cit. on pp. 45, 211).

[WKN08] M. Wieland, P. Kaczmarczyk, and D. Nicklas. “Context Integration for Smart Workflows.” In: *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM ’08)*. IEEE, Apr. 2008, pp. 239–242 (cit. on pp. 45, 211).

[WKNL07] M. Wieland, O. Kopp, D. Nicklas, and F. Leymann. “Towards Context-Aware Workflows.” In: *CAiSE’07 Proceedings of the Workshops and Doctoral Consortium Vol.2*. June 2007 (cit. on pp. 45, 211).

[Wor98] World Wide Web Consortium (W3C). *Extensible Markup Language (XML)*. W3C Recommendation. 1998, p. 16. URL: `https://www.w3.org/TR/2006/REC-xml11-20060816/` (cit. on p. 74).

[WRR08] B. Weber, M. Reichert, and S. Rinderle-Ma. “Change patterns and change support features – Enhancing flexibility in process-aware information systems.” In: *Data & Knowledge Engineering* 66.3 (Sept. 2008), pp. 438–466 (cit. on pp. 49, 211, 212).

[WZN16] X. Wang, Z. Zhao, and W. Ng. “USTF: A Unified System of Team Formation.” In: *IEEE Transactions on Big Data* 2.1 (Mar. 2016), pp. 70–84 (cit. on pp. 38, 196).

[XYWV12] F. Xia, L. T. Yang, L. Wang, and A. Vinel. “Internet of Things.” In: *International Journal of Communication Systems* 25.9 (Aug. 2012), pp. 1101–1102 (cit. on p. 44).

[YL10] S. Yongchareon and C. Liu. “A Process View Framework for Artifact-Centric Business Processes.” In: *On the Move to Meaningful Internet Systems: OTM 2010*. Springer, Oct. 2010, pp. 26–43 (cit. on pp. 54, 55).

[YMMS09] S. Yarosh, T. Matthews, T. P. Moran, and B. Smith. “What Is an Activity? Appropriating an Activity-Centric System.” In: *Human-Computer Interaction: INTERACT 2009*. Springer, Aug. 2009, pp. 582–595 (cit. on p. 56).

[YWLW12] H. Yang, L. Wen, Y. Liu, and J. Wang. “An Approach to Recommend Resources for Business Processes.” In: *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*. Springer, Sept. 2012, pp. 662–665 (cit. on p. 37).

[ZAA07] J. Zhang, M. S. Ackerman, and L. Adamic. “Expertise Networks in Online Communities: Structure and Algorithms.” In: *Proceedings of the 16th international conference on World Wide Web (WWW ’07)*. ACM, May 2007, pp. 221–230 (cit. on p. 38).

[ZC03] Y. Zhou and Y. Chen. “Project-oriented business process performance optimization.” In: *Systems, Man and Cybernetics, 2003. IEEE International Conference on (SMC 2003)*. Vol. 5. IEEE, Oct. 2003, 4079–4084 vol.5 (cit. on p. 48).

[ZCB10] Q. Zhang, L. Cheng, and R. Boutaba. “Cloud computing: state-of-the-art and research challenges.” In: *Journal of Internet Services and Applications* 1.1 (May 2010), pp. 7–18 (cit. on pp. 58, 59).

All URLs were followed on 11.10.2018.