Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis Nr. 0838-010

Recognition of Resource Patterns in Human-centric Processes: A Case Study

Mozi Song

Course of Study:	INFOTECH
Examiner:	Prof. Dr. Frank Leymann
Supervisor:	M.Sc. Celal Timurhan Sungur
Commenced:	October 15, 2015
Completed:	April 15, 2016
CR-Classification:	H.3.3

Abstract

Business experts need to improve business processes by increasing process efficiency and reducing process costs. To achieve this, a common method is to capture a series of repeatedly conducted process activities and their structure, i.e. the business logic of the process, and then enact process execution based on it. However, there exist informal processes, which are human-centric processes that are highly dynamic. Since this approach assumes the existence of predictable business logic of the process, it does not apply for management of informal processes.

The Informal Process Essentials (IPE) model is a modeling approach for informal processes. This model depicts informal processes by documenting resources used in these process. Through this approach, we are able to retain best practice and knowledge accumulated in the processes. Based on this approach, there is also the InProXec method to enable the application of the IPE approach in organizations.

In this thesis work, we want to validate the concepts introduced in the InProXec method using a case study on the jclouds project. To achieve this aim, we introduce the concept of a generic mapping mechanism and an evolving correlation coefficient function. Based on these concepts, we present the Informal Process Discoverer (IPD) service. The IPD service is an implementation of the discovery of IPE models. The test results of the IPD service have shown that the service is successful in discovering the IPE model and giving process recommendations. For example, using an informal process model with includes 7 human resources and 2 GitHub repositories as input, we are able to discover 74 other resources that participate in the process including 65 human resources and 9 Git repositories.

Contents

List of Figures

List of Tables

1 Introduction

In order to gain competitiveness, organizations need to increase efficiency and reduce cost in their development and production processes [BE02]. To provide a baseline for process improvement, it is common for organizations to use *Business Process Management (BPM)* techniques. BPM is a discipline that uses various methods to discover, model, automate, analyze, measure, improve, and optimize business processes [Gar16; Wes07]. It provides many benefits and potential uses such as knowledge retention, program planning and framework for metrics [DM03]. A BPM cycle involves different actors such as business experts and technical experts. In this work, we address BPM actors altogether with the notion *business experts*.

A key concept in BPM is *business logic*, i.e. the series of activities and their structure. In the traditional area of BPM, recurring activities are captured in order to retain business logic. As an example, the OASIS standard executable languages BPEL [Sta07] and BPMN [OMG11] are invented in order to depict these activities in an automatically executable way [Sil11].

However, these modeling approaches assume that processes are *highly structured*. This means that most process instances will follow the exact same business logic [Ros11]. In fact, the execution of these processes are often forced into business logic defined at design time [DHA06]. In reality, there exist *unstructured processes* and *semi-structured processes*, i.e. processes with no predefined business logic, which are typically mostly conducted by humans [MGM+06]. In this work, we address both unstructured processes and semi-structured processes as *informal processes*. In informal processes, human actors make decisions based on their knowledge, experience, the process situation, and other subjective factors. This makes informal processes ad-hoc and difficult to predict beforehand. For example, decisions of a bug-fixing process depend on the nature of the reported bug, the working habits of the human actor assigned to the task, the time limit, guidance, and other conditions given by the assignee's director, etc. Process modeling of these processes must be carried out after understanding the essence of informal processes. Otherwise, the most likely result will be continuing unsatisfactory practice due to outdated knowledge on the processes [GK02].

Due to the fact that informal processes are semi-structured or unstructured, it will not be beneficial to depict informal processes by capturing their business logic, i.e.

trying to discover the activity sequence and structure in informal processes. This is not only because they are highly dynamic, but also due to the fact that it would be expensive to create a well-defined informal process which is highly dynamic [LR00; Nur08]. Moreover, defining the business logic may retrain the flexibility of the process and thus possibly decrease human creativity [SBBL14a]. Therefore, we need a new approach of process modeling in order to simplify automatic process execution but still retain best practice and knowledge.

To address the above mentioned problem, Sungur et al. [SBBL14b] presented a resource-centric modeling approach called *Informal Process Essentials (IPE)* based on requirements driven from literature and a set of interviews. An IPE model depicts informal processes by documenting resources used in those processes, and contains context definitions such as process intentions, process sub-intentions, organizational capabilities, resource relationships, etc. In other words, with an IPE model, business experts are able to model the process without describing *how* processes goals are reached, but instead *what* is needed for reaching the process intention. These resources include not only human performers, but also other types of resources such as IT resources, knowledge resources, and material resources. By describing the existence and capabilities of these resources, the business logic of the process is implicitly defined, since the resources contain best practice that are available for future use [SBBL14a]. The IPE model defines a new paradigm which is opposed to classical business modeling approaches, but still aims at retaining sufficient historical process information required for process improvement. Although the model does not include explicit activity sequences, it suffices reusing best practice accumulated during execution of informal processes.

To enable automated initialization and creation of informal processes based on the IPE modeling approach, Sungur et al. presented the InProXec method [SBLW15]. Several crucial concepts of the InProXec method during discovery phase include *involved resources, relevant resources, relevance relationships* and *relevant capabilities*. In short, involved resources are resources known to participate in the process, whereas relevant resources are the ones which interact with involved resources and other relevant resources in an informal process, such as an external expert in a software development process. A relevance relationship is a notion which represents an association of a resource or a capability on the informal process model. A relevant capability is a capability of an involved resource or a relevant resource that is not documented in an informal process. For more details, please see Chapter 3.

1.1 Problem Definition and Contributions

Based on the IPE model, *process mining* techniques can be used for informal process discovery. Process mining is a way to analyze processes based on the event logs [AG07]. Here, event logs refer to logs recording interactions among human resources as well as between human resources and other types of resources. From the event logs, relevant resources and relevant capabilities can be inferred. With these concepts, we want to construct a *process recommendation*, i.e. a process model containing the discovered resources and capabilities that can be used to achieve a similar process goal as of the process being mined.

In this thesis work, we present the *Informal Process Discoverer (IPD)* service as an implementation of the discovery of IPE models. The main contribution of this thesis is the validation of concepts introduced in the InProXec method using a case study on the jclouds project. To achieve this aim, we also introduce a concept of a generic mapping mechanism and an evolving function to describe the level of relevant capabilities. These concepts will be later used in the IPD service for deduction of relevant resources and relevant capabilities.

1.2 Thesis Outline

The remaining master thesis document is structured in the following way:

- *Chapter 2-Motivating Scenario:* This chapter presents a motivating scenario based on which the case study is built.
- *Chapter 3-Fundamentals:* The basis of the IPD service–the *Informal Process Essentials (IPE)* approach presented in [SBBL14a] is introduced in this chapter. Then, an apporach for automating the enactment of informal process–the InProXec method from [SBLW15]–is described.
- *Chapter 4-Case Study:* This chapter focuses on the architecture and implementation of the IPD service. Here, the mechanism of mappers as well as an evolving function for depicting capability levels are introduced. After that, the results of process recommendations on the motivating scenario are given.
- *Chapter 5-Related Work:* In this chapter, we have chosen and analyzed other efforts that focus on semi-structured or unstructured processes, or similar directions such as social network discovery, and expertise discovery.

- *Chapter 6-Conclusions and Future Work:* In the last chapter, a summary of this thesis and possible future directions are given.

This document also contains an appendix for look-up:

- *Appendix A-List of Acronyms:* This appendix contains all the acronyms used in this document.

2 Motivating Scenario

We have chosen the Apache jclouds project[1] as our motivating scenario. The jclouds project makes a very good scenario for making IPE recommendations of informal processes for the following reasons. Firstly, there exist different types of process resources, including human performers, such as Java developers, and IT-resources, such as GitHub repositories. Some of them are known at project creation time, and the others participate in the project later, such as external contributors. This gives us the motivation of finding them through analysis as relevant resources. Secondly, due to the nature of the jclouds project that it is designed for multiple cloud vendors, many Git repositories are created, which provide sufficient information of resource interactions for the analysis.

The jclouds project aims at providing open-source unified Java libraries and thus preventing vendor lock-in. As traditional for the Apache Software Foundation (ASF)[2], the core development group of the project is called the Project Management Committee (PMC), the members of which are able to make decisions on changing the sources code or documentation. Since admission to PMC is strictly regulated, e.g. the developer needs to be nominated by a PMC member and make at least six months of active contributions, there is a much larger group of external contributors outside of PMC [BGD+07]. In fact, as an open-source project, the jclouds project relies on external contributors since they provide valuable software innovations on their corresponding areas of expertise.

On GitHub, a jclouds repository[3] is set up, to which contributors can make contributions by creating pull requests. Since jclouds is designed for multiple services on platforms provided by different cloud vendors, several repositories beside the main repository are created for various purposes. For example, the jclouds-karaf repository[4] is provided for installation of jclouds inside Apache Karaf, whereas the jclouds-examples repository[5] contains example code of using jclouds.

[1] https://jclouds.apache.org/
[2] http://www.apache.org/
[3] https://github.com/jclouds/jclouds
[4] https://github.com/jclouds/jclouds-karaf
[5] https://github.com/jclouds/jclouds-examples

During the process, several sub-processes are repeatedly performed, such as developing the code, making innovations to the project and launching technical discussions with other people. These reoccuring activities expose certain capabilities that will possibly be necessary in future processes with similar intentions, such as developing another software. However, simply recording the activities and repeating them in the next execution of an informal process with similar process goals would not be beneficial, since informal processes are highly dynamic. Fortunately, a part of these activities are conducted as interactions between two resources and are thus automatically recorded, such as the log of events on GitHub. Through analyzing these event logs, we can discover relevant resources and relevant capabilities of the informal process. Furthermore, we need a way of addressing the relevance of relevant resources and relevant capabilities, so that we can make a process model recommendation including a sorted list of them. To achieve this aim, we have proposed the concept of *relevance relationships* and *correlation coefficients*. The first concept associates relevant resources and relevant capabilities with the mined informal process, while the second one denotes the level of relevance of these.

An example of initial process structure using Winery [KBBL13] is shown in Fig. 2.1. In this scenario, we assume that a sub-process goal of “fixing a bug” is specified. The structure of the given process instance shows the runtime process structure at a given time point, and provides information about resources such as resource location, resource ID, etc. The process structure includes four human performers and two GitHub repositories, i.e. involved resources. Based on the involved resources, the problem is to discover other resources that take part in the process at process runtime. At this time, we also have certain information about resource capabilities, e.g. Ignasi Barrera has code development capability. On the other hand, the capability of Brian McCallister is unknown. This leaves us the opportunity of discovering the relevant capability provided by him. With the IPD Service and complementing concepts such as relevance mappers and an evolving function for relevant capability correlation coefficients, we want to demonstrate that based on the InProXec method, it is possible that we retain valuable knowledge and experience from past process executions, such as relevant resources and capabilities, that are helpful in increasing process efficiency.

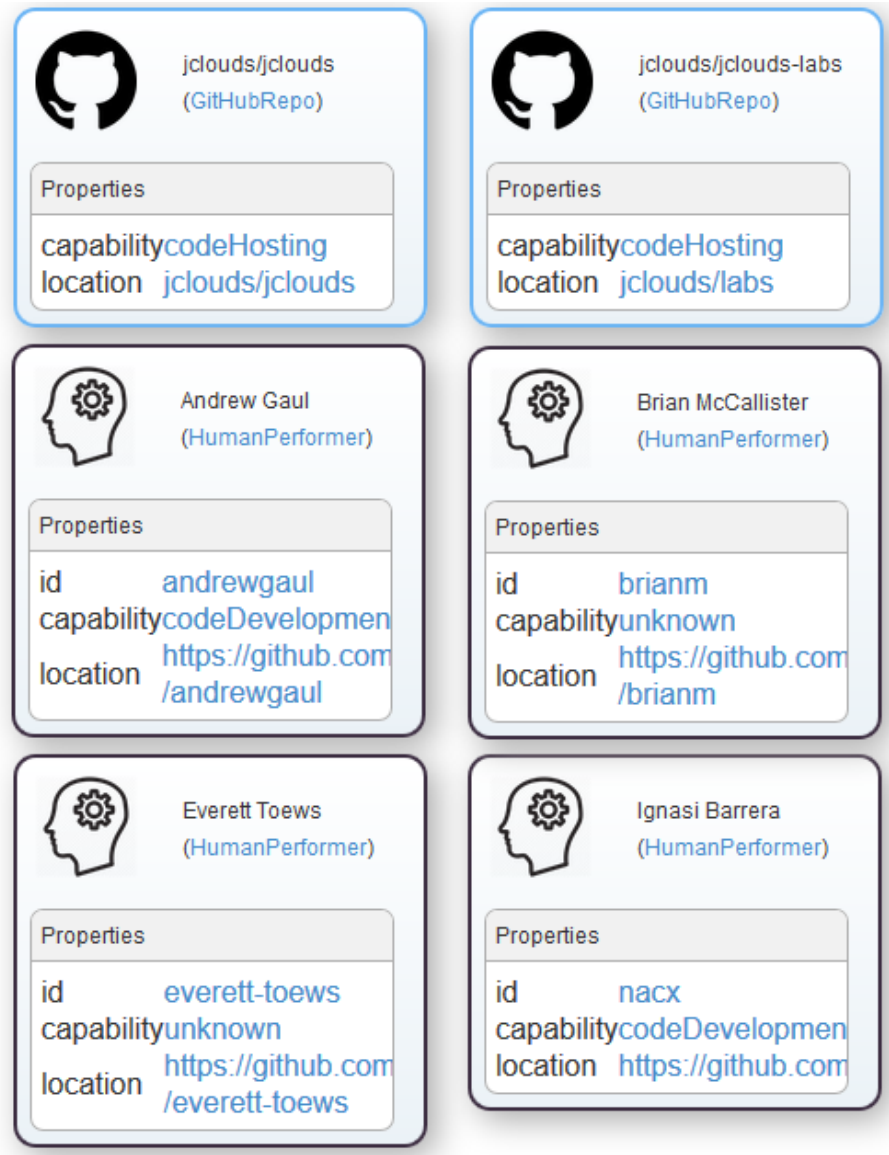

Figure 2.1: An example initial process structure from the motivating scenario with two GitHub repository resources and four human resources.

3 Fundamentals

In this chapter, we give fundamental information based on which the Informal Process Discoverer service is developed. Section 3.1 introduces properties of informal processes defined in [SBBL14a]. Based on these properties, we further discuss informal process modeling. Section 3.2 introduces Informal Process Essentials defined by [SBBL14a] which is necessary for depicting the resource-centric model for informal processes. Then, Section 3.3 describes the approach that enables the application of the IPE approach in organizations–the InProXec approach raised by [SBLW15]. Here, we provide an extended version of this approach to support informal process discovery. For this aim, the concepts of *relevance relationships*, *resource analyzers*, *relevance mappers*, and *informal process recommenders* are proposed. Finally, since the TOSCA specification is utilized in the represent resource models of informal process models and instances, Section 3.3.1 gives a brief introduction on TOSCA. Finally, Section 3.4 gives a brief summary on concepts introduced in this chapter.

3.1 Properties of Informal Processes

An informal process can be either a standalone process or a sub-process of a structured business process. The most important feature of an informal process is that it is human-centric with activities that can not be predicted, or predicting them does not add more value than its costs. In other words, the activities in an informal process are conducted ad-hoc based on human decisions [SBBL14a]. In order to lay foundation for developing Informal Process Essentials, Sungur et al. concluded in [SBBL14a] that informal processes have the following characteristic properties:

i Implicit Business Logic: Due to the unpredictable nature of informal processes, it is often impossible or too expensive to define business logics for them.

ii Various Relationships among Resources: In processes with complex problems, networked humans and other softwares are utilized. These resources typically make interactions with each other or depend on each other, and must work in a corporative way. Their interrelationships could be utilized by process experts in analyzing the process as well as making process recommendations.

iii Participation of Resources in Several Processes: For each resource in a certain informal process, it is possible that they simultaneously participate in, or they have historically been involved in, other processes.

iv Resource Variation: During execution time, one or more resources can leave or participate in the process. In other words, the set of resources do not remain static after initialization of the process. One example is that in a software development process in a company, the development department may request the participation of experts in another department who expert in the basis framework utilized for the new software.

In informal processes, technical knowledge and experience of humans are used to reach process goals. Time, personal energy and financial resources are spent on processes which may include certain repetitive tasks. During their work, individual performers accumulate best-practices that can be used in future processes with similar goals. Thus, it would be unfortunate if they could not be utilized again. Since activity-based modeling is not suited for such processes, we change our view on process modeling. Instead of describing *how* goals are achieved, we define *what* is needed to achieve these goals, i.e. we implicitly define business logic by defining resources used in the process. By including these resources, we include their experience and capabilities in a new process and decrease the cost of modeling and executing informal processes, since business logic is not needed.

3.2 Informal Process Essentials

Based on the properties of informal processes introduced in Section 3.1, Sungur et al. designed the concept of Informal Process Essentials (IPE) [SBBL14a]. This concept includes the essential information used to describe an informal process. Thus, this section gives an introduction on IPE.

Fig. 3.1 shows a meta-model of the IPE approach. In this approach, modeling of an informal process starts with defining the process *intention*. This specifies the final goal of the process, e.g. "develop a website for online-shopping", or "customize the software for special needs". In order to complete process goals, intentions can be refined by defining sub-intentions. Apart from process intentions, as a part of the organizational strategy, *initial context* and *desired final context* can be also specified. Initial context describes the starting state of the process, such as "a demand for a new online-shopping website" or "customer needs for special features in the software". The desired final context defines the expected state to be reached at the end of process execution, such as "successful development of the website" or "implementation of special features for customer needs".

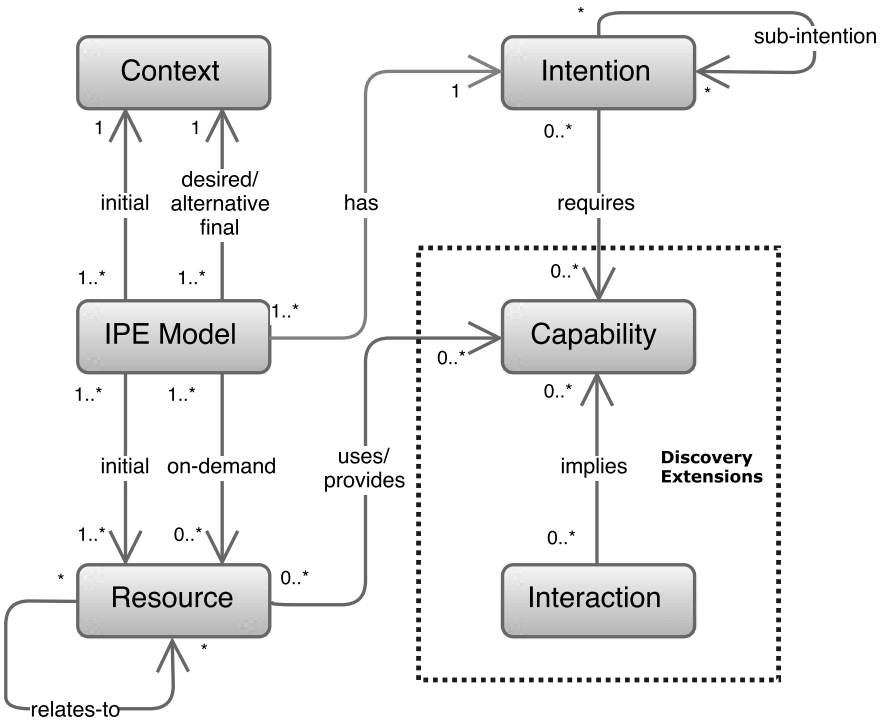

Figure 3.1: The conceptual Meta-model of Informal Process Essentials adapted from [SBBL14a].

Instead of a successful final context, an *alternative final context* could also be defined in case of process failure, such as "unsuccessful efforts in website development".

According to the defined process strategy, *IPE Resources* that are viewed as beneficial for the process goal could be engaged into the process, such as an online Wiki or a Java developer. Apart from being integrated into the process at process initialization time, IPE Resources could also be engaged during process runtime, which corresponds to the flexibility feature of informal processes. Each IPE Resource may provide one or more *capabilities*, such as Java development capability. In case of informal process discovery, i.e. the discovery of relevant resources and relevant capabilities, these capabilities can be inferred during process execution by mining resource interactions. For example, if a human performer pushes to a Git repository, it may indicate that the human has code development capability. A database with IPE Resources which are associated with such capabilities can be previously set up manually, and can be populated during process discovery. Whenever a new process intention is given, we can make decisions on which IPE Resources to use by resolving capabilities required by the process intention into corresponding IPE Resources which provide these capabilities.

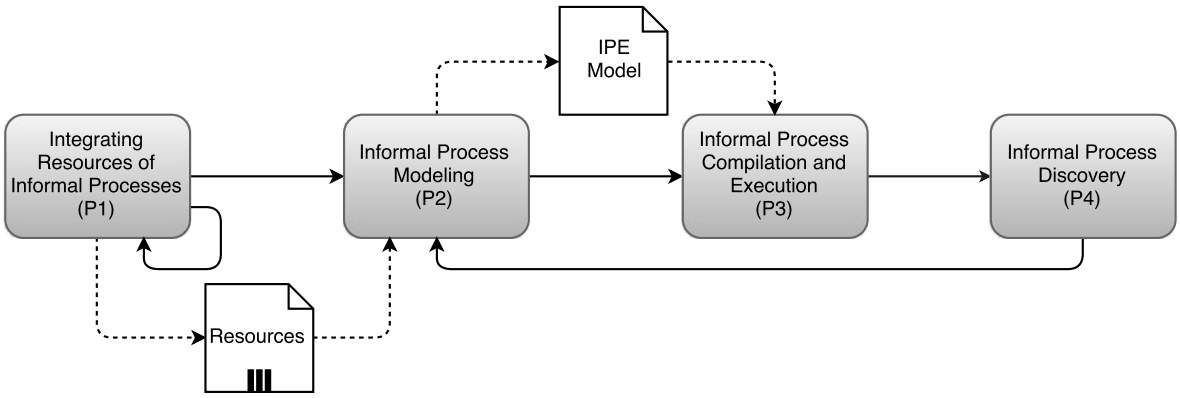

Figure 3.2: The Extended InProXec system pipeline, adapted from [SBLW15] with the adding of P4

3.3 InProXec Method

InProXec is a method presented in [SBLW15] to automate informal process execution. In this method, three main phases are designed. The main content of these three phases are integration of resources, informal process modeling and compilation, and informal process compilation and execution. Here, we present an extended InProXec approach with the addition of a fourth phase–informal process process discovery. Fig. 3.2 shows an overview on the different phases of the extended InProXec method. The following paragraphs will introduce the contents of each phase.

The first phase (P1) aims at setting up the execution environment for informal process executions. Firstly, resources involved in the process should be identified. Here, four main resource domains are possible: human resources, IT resources, knowledge resources and material resources. After that, the identified resources are integrated into the modeling and execution environment. For that aim, elements such as *retrieval services* are used. Retrieval services are able to provide information about resources and their properties, such as a social networking service which lists experts in different domains and their detailed information. For each retrieval service, experts use *resource engagers*, i.e. runnables such as BPEL processes, to automate the retrieval of resources. To provide runtime environments for resource engagers, *control services* are designed, for example, a BPEL engine. Resource engagers work in the environment provided by control services to acquire resource, e.g. a BPEL process runs in a BPEL engine and contacts a human expert using the API of a retrieval service, such as XING, providing an invitation to participate in a certain informal process and details of the process. Integrating different Control Services into execution environments may mean that the environment must be changed, since different Control Services provide different APIs. Thus, experts hide Con-

trol Services behind a *execution environment integrators*, a wrapper service that unifies access to different Control Services. We also build *domain managers* wrappers around retrieval services that deliver the resource definitions and a resource engager, which is then deployed on an execution environment integrator. Each execution environment integrator uses a control service to deploy a resource engager. After P1, resources of interest in different domains are collected and presented to business process experts for next phase.

In the modeling phase (P2), business experts model informal processes using the resources collected in phase 1. In the process initialization and execution phase (P3), the process model should be initialized. After initialization, an *informal process instance* should be created. Compared to an Informal Process Model, an Informal Process Instance contains additional meta-data describing runtime information of the process such as process start time. An Informal Process Instance refers to a real instance of a process, whereas an Informal Process Model can refer to resources that do not yet exist. The resource instances in an Informal Process Instance contain *instance descriptors* which make it possible to track or communicated with the certain resource. For example, a resource instance for a Java expert may include his full name, his Email address, his personal page, and his ID on certain code repositories, etc. After initialization of the given informal process, human performers, with the aid of other resources such as a MediaWiki[1] resource, enact the execution of informal process by working automatically towards the process intention. At this time, changes may be made to the process model by business experts in order to adapt to new situations.

To make it possible for informal processes to be modified according to changing situations at process runtime, we extended the InProXec method by a fourth phase–informal process discovery (P4). In this phase, the IPD service implemented in this work is utilized. By analyzing existing process interactions, the IPD service makes discoveries of resources that participate in the process after process initialization, as well as capabilities that are exposed during process execution. With the recommendation constructed by the IPD services, business experts can make adaptations to the process model such as adding resource instances, modifying organizational capabilities, etc. In this way, the InProXec method becomes evolutionary, i.e. by the help of the recommender, the process converges to the real world scenario gradually [SBBL14a]. In order to discover process changes, we introduce the concepts of *relevance relationships*, *resource analyzers*, *relevance mappers*, and *informal process recommender* in the following paragraphs.

An important phase of informal process discovery is the discovery of relevant resources and relevant capabilities. Here, we need a "confidence level" to depict the extent on which a relevant resource is related to the process being mined. Also, a "capability level"

[1] https://www.mediawiki.org/wiki/MediaWiki

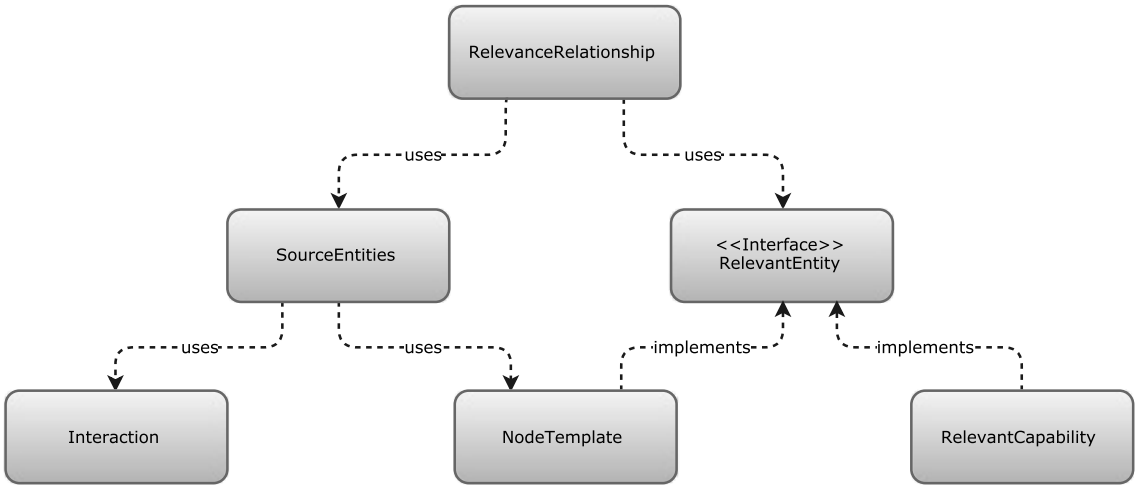

Figure 3.3: The concept composition of Relevance Relationship.

describing how much a certain relevant capability is exposed in a process should be included. In order to include this information in a concise way, we define the concept of *Relevance Relationships*.

Fig. 3.3 shows the composition of the Relevance Relationship concept. A Relevance Relationship consists of a Relevant Entity an array of Source Entities. The Relevant Entity can be either a relevant capability or a relevant resource. The Source Entities could either be a list of interactions or a list of Node Templates. Moreover, a Relevance Relationship has a field called *correlation coefficient*, which depicts the "confidence level" or "capability level" mentioned above with a float value.

For informal process discovery, we also need components which can retrieve and "understand" interactions in event logs. For this aim, we first present the concept of *resource analyzers*. A resource analyzer is a component which retrieves interactions from different IT resources. For example, for a GitHub repository, we have the Git resource analyzer which sends requests to the Git API server and retrieves a list of interactions. To "understand" what these interactions mean, we present the concept of *relevance mappers*. A relevance mapper uses interactions and existing relevance relationships to create new relevance relationships using interaction mappings. To indicate the level of confidence of a certain mapping, mapping correlation coefficients are introduced.

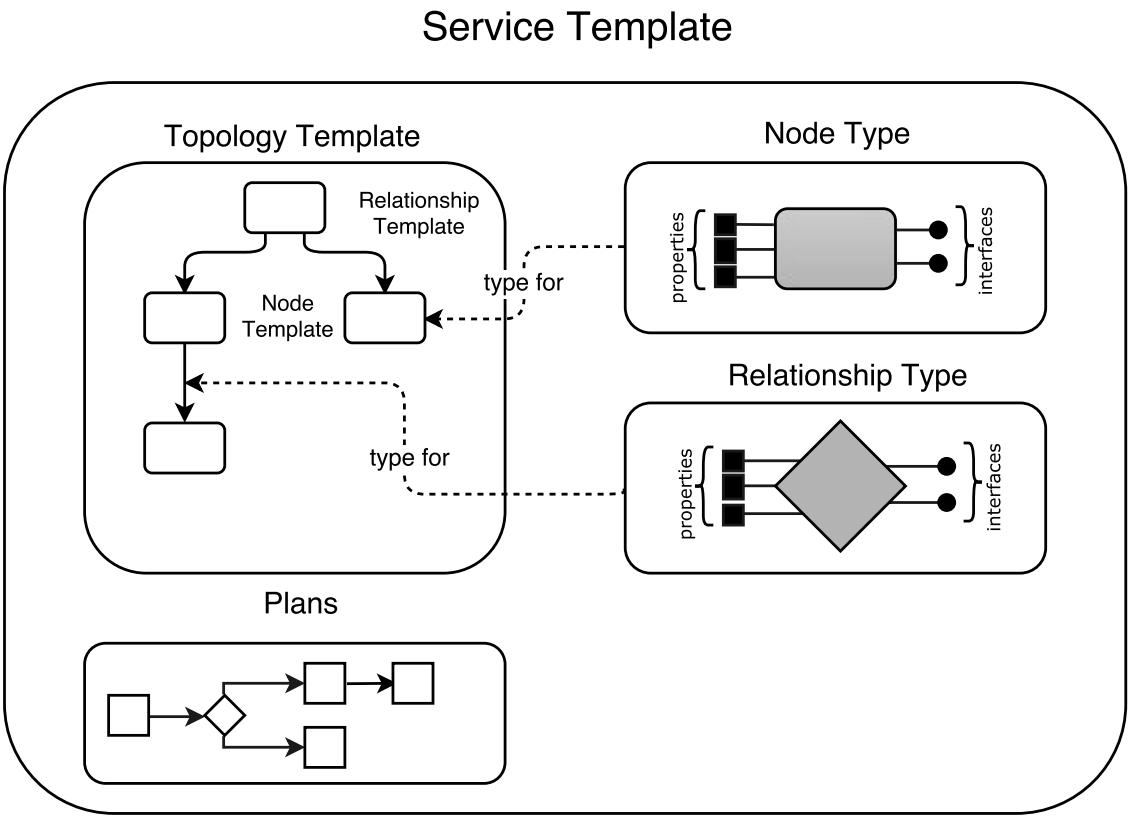

Figure 3.4: The structure of the TOSCA Service Template adapted from the OASIS Topology and Orchestration Specification for Cloud Applications [OAS13].

3.3.1 Utilization of TOSCA

In our work, we adopt the Topology and Orchestration Specification for Cloud Applications (TOSCA)[2] from OASIS as the basis of our resource-centric informal process modeling. TOSCA proposes an XML-based modeling language for cloud application structure modeling with directed graph topologies [BKL14]. By providing a standardized way in defining composite applications and their management [BBLS12], TOSCA offers solutions for portable deployment and management of cloud applications. Thus, by using TOSCA in our work, we ensure unification in modeling and managing different informal processes. Here, we give a brief introduction on TOSCA.

[2]http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

As shown in Fig.3.4, a cloud service is an instantiation of a *service template*. A Service Template is composed of two main concepts: *topology template*, which defines the service structure, and *plans*, which provides information used to manage the service during its lifetime. A Topology Template includes a certain number of *node templates* and *relationship templates*. Node Templates are real-life instantiations of *node types*. A Node Type depicts a component by specifying its properties and deployment operations. Similarly, a Relationship Template is an instance of a *relationship type*, which defines its source and target element as well as the relationship direction. Here, Node Types and Relationships Types are designed to be reference and reused, whereas Node Templates and Relationship Templates are unique and correspond to real life occurrences.

The TOSCA specification shows many similarities as our resource-centric model of informal processes. Firstly, the resources in our model can be seen as nodes in TOSCA. A Node Type can be used to represent a certain resource type, such as a human performer or a Redmine instance, etc. It depends on the process expert on which granularity the Node Types are defined. After informal process initialization, the Node Types can resolve into Node Templates which include instance descriptors of the certain resource. For example, we can have a Node Type "human actor" and multiple human performers such as Java developers, project managers and external contributors as instances of it.

In our work, concepts needed to model informal processes are defined based on TOSCA. We utilize certain types and templates in TOSCA, such as Node Types and Node Templates. Based on these, we create our customized concepts for informal processes, such as Strategies, Informal Process Definitions, etc.

3.4 Summary

In this chapter, fundamental information of the Informal Process Discoverer is introduced. We introduced the IPE modeling approach and InProXec method based on the properties of informal processes. For recapture on the key concepts introduced in this chapter, we give a concept definition overview in Table 3.1.

Name	Definition
Business Process Management (BPM)	A discipline that uses various methods to discover, model, analyze, measure, improve, and optimize business processes.
business logic	The series of activities and their structure in a process.
informal processes	Processes with no predefined business logic.
involved resources	Resource known to participate in the process.
relevant resources	Resources inferred to take part in the process.
relevant capability	A capability of an involved resource or a relevant resource that is not documented in the process.
Informal Process Discoverer (IPD) service	An implementation for discovery of relevant resources and relevant relationships based on involved resources with two recursions.
relevance relationship	a concept associating relevant resources and relevant capabilities with the informal processs
resource analyzers	components which retrieve interactions from different IT resources
relevance mappers	components which use interactions and existing relevance relationships to create new relevance relationships using interaction mappings

Table 3.1: An overview of key concepts introduced in Chapter 3.

4 Case Study Of The jclouds Project

In this chapter, we present a case study on the jclouds project as a proof-of-concept of the extended InProXec approach, as well as the necessary concepts needed for the case study. As an implementation of the extended InProXec approach, we present the *Informal Process Discoverer (IPD)* System. In this system, we adopt an iterative method with depth=2 to find relevant resources and relevant capabilities, i.e. in the first iteration, we analyze interactions enacted by involved resources for relevant resources; in the second iteration, we treat the resulting relevant resources of the first iteration as involved resources, and discover new relevant resources based on these. The design of the second iteration brings two benefits. Firstly, we are able to discover more relevant resources in the informal process than using only one iteration, which means a larger portion of process information is utilized. Secondly, if the set of involved resources does not provide sufficient information to mine the process, e.g. only one human performer is known to participate in the process, this approach compensates for the incompleteness by utilizing the relevant resources resulting from the first iteration.

In this chapter, we start with describing the core system design of the Informal Process Discoverer service, then the implementation details and case study results will be discussed.

4.1 The Core System Description of the Informal Process Discoverer

In this section, we introduce the Informal Process Discoverer service. Firstly, Section 4.1.1 presents a meta-model of the IPD service in order to depict core concepts used in the service and logical relationships among them. After that, Section 4.1.2 introduces a process diagram indicating the different steps involved during process discovery. To help readers better understand the system design, Section 4.1.3 describes a class diagram with core interfaces in the system code. In Section 4.1.4, a generic mapping mechanism utilized in the IPD service is introduced. Finally, in Section 4.1.5, we explain how the iterative formula of correlation coefficient is designed.

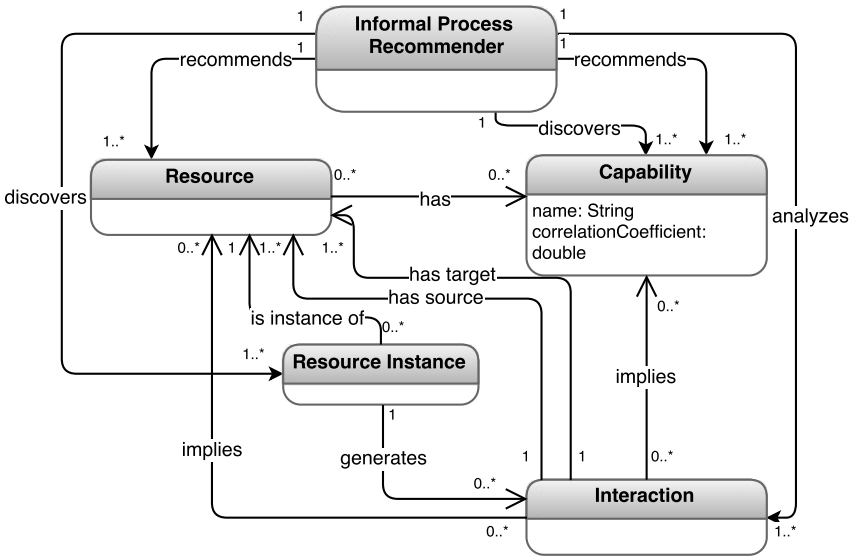

Figure 4.1: Meta-model of IPD System

4.1.1 Meta-Model of the Informal Process Discoverer System

The meta-model of this system is shown in Fig. 4.1. In every informal process, there are certain known resource types, such as internal software experts, bug-tracking services, etc. These resources are known as *involved resources* in our work. At execution phase of the process, resource instances for these resource types are generated. The resources unknown at process initialization phase, i.e. the *relevant resources*, are also instantiated. To reach the process goal, resource instances will make interactions such as sending emails, leaving comments on GitHub repositories, and modifying online Wiki services for the project, etc. In our system, these interactions will be analyzed and interpreted, with the aim of finding unknown resources, i.e.*involved resources*, and capabilities, i.e. *relevant resources* that are implied by interactions. In fact, in our assumption, a database containing available organizational resources and their capabilities should be at hand. Thus, by finding out the most crucial capabilities in the process, we in fact make recommendations of most important resource types, since the capabilities can be mapped to resources with the above mentioned database.

4.1.2 Different Steps of the Informal Process Discoverer System

As indicated in Fig. 4.2, there are two inputs of the recommender system (see the input data of P_1): an informal process model and one or multiple informal process instances. In P_1, the resources specified in the input *informal process instance* are

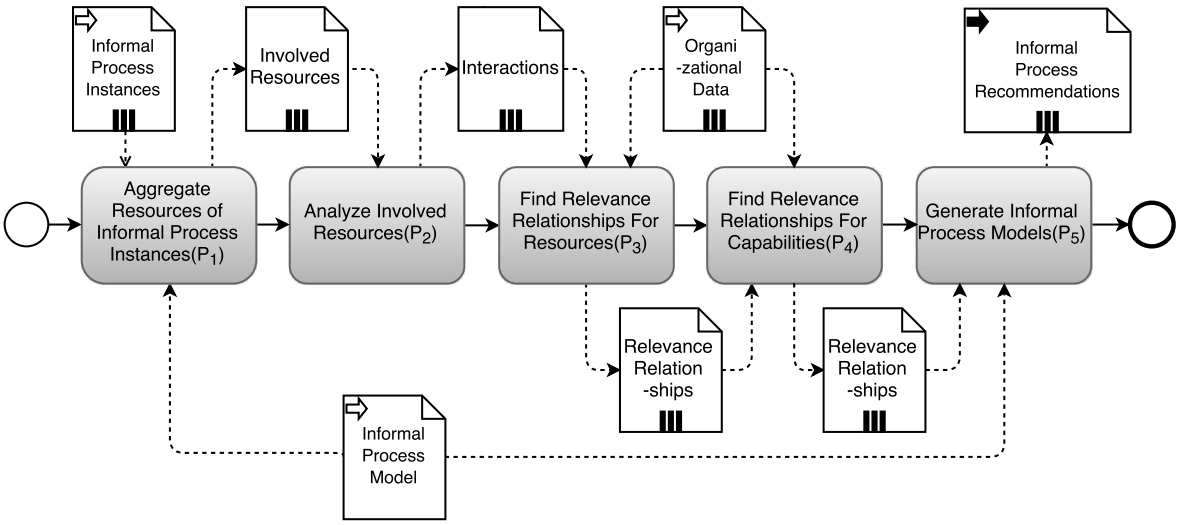

Figure 4.2: Different Steps of the Informal Process Discoverer System.

extracted and aggregated. In P_2, these resources are analyzed as *involved resources* for *interaction*s. An *interaction* includes the source- and target resources and the interaction type. For example, for the GitHub repository event "Ignasi Barrera makes a pull request to GitHub jclouds/jclouds", a new *interaction* with the human actor `Ignasi Barrera` as the interaction source, the IT-resource `GitHub jclouds/jclouds` being the interaction target, and `"pullRequest"` as the interaction type will be generated.

Injected with the *interaction*s resulting from P_2, the first level of interpretation takes place in P_3. This interpretation recognizes resources that make interactions with *involved resources*, i.e. *relevant resources*, and generates *relevance relationship*s for these resources. For readers' understanding, a *relevance relationship* represents resources or capabilities implied by *interaction*s. Either way, it contains a *correlation coefficient* indicating the level of relevance of the corresponding resource or capability. Here, the *correlation coefficient* of a *relevant resource* will be increased whenever an interaction containing the resource (as resource *or* target) occurs, indicating that the *correlation coefficient* is in theory proportional to the frequency of interactions made by the *relevant resource*.

The *relevance relationship*s generated in P_3 are passed on to a second-level interpreter–the capability interpreter–in P_4. Here, *capability mapping*s are read from organizational data. *Capability mapping*s are interaction-capability data pairs with *weight*s indicating how important the interaction type in sight is to the corresponding capability. For example, a `push` interaction to a Git repository may imply `code development capability` and `code hosting capability`, in the angles of the human actor and the Git repository. In this step (P_4), similar as in P_3, we generate *relevance relationship*s for each *relevant capability*.

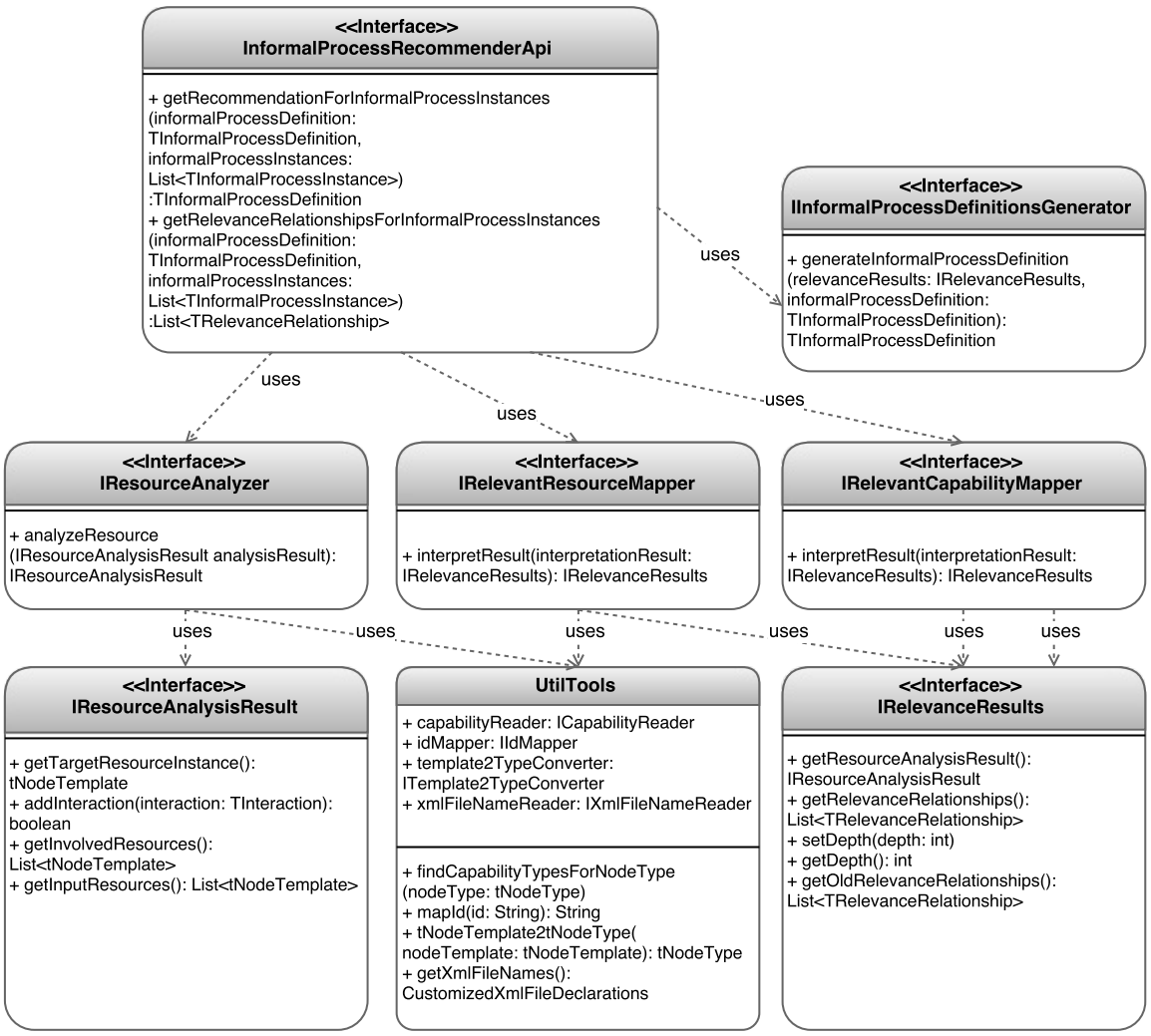

Figure 4.3: Illustration of the Core System

4.1.3 Class Definition of the Informal Process Discoverer Service

The main API in the system is `InformalProcessRecommenderApi`, as shown in Fig. 4.3. It has two main functions using the same set of input parameters: a `TInformalProcessDefinition` object and a `list` of `TInformalProcessInstance` objects. As explained in Section 4.1.2, the first parameter depicts the informal process model, while the second represents the "execution time" informal process model instances. The first method returns an updated `TInformalProcessDefinition` object as recommenda-

tion, and the second one returns a `list` of `TRelevanceRelationship` objects. If detailed information of the interpretation results are needed, the second method can be used.

We designed different interfaces for steps P_2 - P_5 in Fig. 4.2. In P_2, `IResourceAnalyzers` are called to launch analysis on the involved resources. In P_3, `IRelevantResourceMappers` are used to interprete the interactions resulting from P_3 for relevant resources. In P_4, `IRelevantCapabilityMappers` are used to interprete interactions for relevant capabilities. In P_5, `IInformalProcessDefinitionsGenerators` generate process definitions from relevant resources and relevant capabilities.

There are two result APIs that store analysis and interpretation results and pass them through different analyzers and interpreters: `IResourceAnalysisResult` and `IRelevanceResults`. The first unified interface provides a way for analyzers to be called in the system to work in a random order. Each analyzer reads information from the interface, modifies it, and then passes it to the next analyzer. The `IRelevanceResults` interface works similarly for interpreters.

As shown at top-right of the figure, an implementation of the `IResourceAnalysisResult` interface contains a target resource instance, a list of interactions, a list of input resources, a list of involved resources and a list of to-be-ignored resources. The target resource instance specifies the resource to be analyzed. If it is a human actor, it will automatically be ignored by `GitHubAnalyzer` and will be crawled by `HumanResourceAnalyzer`. Similarly, if it is a Git repository instance, it will be neglected by `HumanResourceAnalyzer` and crawled by `GitHubAnalyzer`. The crawling process will update the interaction list, which is the most important field in `IResourceAnalyzer`. The list of input resources will be later used in interpreters in deciding whether a resource is an *involved* or *relevant* resource. The *involved* resources will then be added into the list of involved resources. The list of to-be-ignored resources are used in iterative analysis where the iterative depth is larger than 1. For example, in the second level of iterative analysis, involved resources of the first level analysis should be neglected because interactions containing these resources are already analyzed and interpreted in the first iteration. Thus, these resources should be added to the list of to-be-neglected resources in the second iteration. For more details, please see Section 4.2.

4.1.4 The Generic Mapping Mechanism

To support the relevance mappers, i.e. relevance resources mappers and relevant capability mappers, a generic mapping mechanism is defined. The mapping mechanism consists of two levels: *resource mapping* and *interaction mapping*. A resource mapping maps aliases of human resources to full names of them. In the IPD service, we manually set up a database defining this mapping. For example, the GitHub ID “nacx” will be

mapped to the human resource with full name “Ignasi Barrera”. An interaction mapping specifies relevant capabilities indicated by different interaction types. This is based on the business expert’s understanding of the informal processes being analyzed. An interaction type may indicated one or multiple relevant capabilities. For example, to support the capability mapper of GitHub interactions, we define the mapping of GitHub events. In our definition, a push event indicates the code development capability from the event initializer, and an `issue` event implies a `project coordination and discussion capability`. Moreover, since these events all happen in GitHub, each event type will indicate a code hosting capability from the GitHub repository.

To indicate the confidence of deductions, all mappings are weighted with a float value between -1 and +1. A weight of a negative value indicates irrelevance of a capability. A high weight value means that we are relatively confident in this certain mapping. The *correlation coefficient*s, i.e. relevance levels, of relevant capabilities, will be increased when an interaction indicating this capability is detected. Here, the *weight* values play a crucial role in deciding the correlation coefficients, as they control the increasing speed of correlation coefficients. An interaction of weight value 0.5 results in a smaller increase than that of weight value 0.8 on a relevant capability (see Section 4.1.5 for more details on correlation coefficients). As a result, the weight values need to be designed carefully. To illustrate the importance of weight values, we give a small example. In a Git repository, both the `watch` and `fork` interactions may imply `software usage capability`, with the former’s weight being 0.5 and the latter’s being 0.7, because rather than `watch`ing, `fork`ing a repository may indicate with a higher possibility that the actor is reusing the code. What’s more, emphasis on a certain relevant capability due to pre-knowledge of the process can be expressed by assigning it weights with relevant higher values than the other capabilities. For example, the weight values of interactions indicating `bug fixing capability` may be much higher than the other capabilities, because the process analyzer has the information that the informal process intention is about fixing a bug. Thus, he adjusts the interpretation by stressing the corresponding capability. For more mathematical details about the evolving formula of *correlation coefficient*s, please see Section 4.1.5.

4.1.5 Design Of The Evolving Correlation Coefficient

Since capabilities are exposed through interactions, theoretically, the most radical way of recognizing capability levels, i.e. on which extent the capability in question is “needed” in this process, would be recording how frequently the capability is exposed through interactions. In order to depict this value, we use *correlation coefficients* which are double values in the interval of [-1.0,+1.0]. When we iterate through `tRelevanceRelationship` objects, we iterate through the interactions recorded in these objects and the more

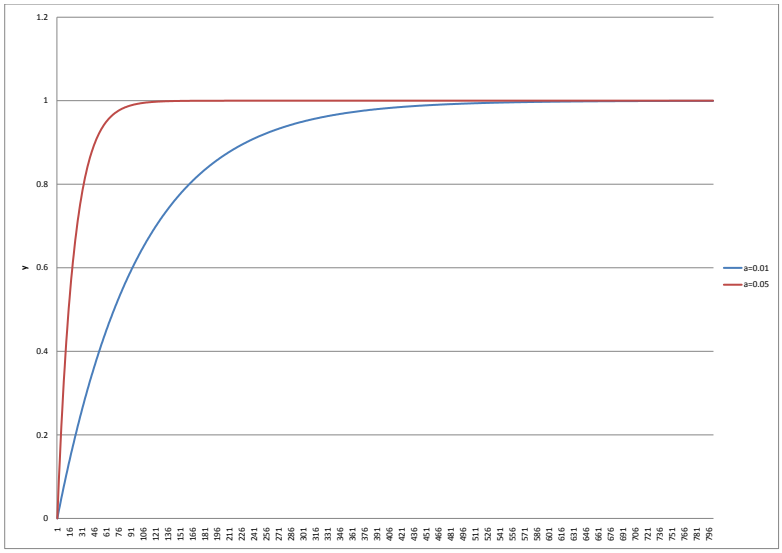

Figure 4.4: The function $y = 1 - e^{(-ax)}$ when *a=0.05* and *a=0.01*

interactions we find pointing to a certain capability, the bigger the correlation coefficient of this capability should be. What's more, user-defined weights and the depth of analysis iteration *d* should also be taken into consideration. If we denote the number of relevant interactions(i.e. interactions that imply a certain capability) as *x*, the weight as *w*, the depth as *d* and the correlation coefficient as *y*, the designation of the formula *y=f(x,w,d)* should be based on five principles:

1. When $x = 0, y = 0$;

2. When *x* approaches $+\infty$, *y* approaches +1.0;

3. The formula should be easily transformed into a recursive formula based on Δx;

4. User-defined weight *w* effects the convergence speed of the formula in the sense that with a smaller weight *w*, *y* approaches +1.0 and -1.0 more slowly;

5. Iterative process analysis depth *d* effects the convergence speed of the formula in the sense that with a bigger depth *d*, *y* approaches +1.0 and -1.0 more slowly. i.e. Capabilities found on second or third iterations of process discovery are not as relevant as those found on the first one.

We designed a function which satisfies the above principles:

$$y = 1 - e^{\frac{-awx}{d}}, a > 0$$

The Fig. 4.4 shows the function when *w=d=1*. The convergence speed to 1.0 can be manipulated by changing the factor *a*. By increasing *a*, *y* approaches 1.0 faster. In

the motivating scenario, we have about 300 interactions for one GitHub repository. When *a=0.05*, the correlation coefficient concludes to 0.8 when there are 33 relevant interactions, which is by experience too tolerant and fast. We need the evolving speed to be slower, so that a difference between a relevant capability with 33 interactions and that with 80 interactions appears evident enough on the correlation coefficient values. Thus, *a=0.01* is chosen. Here, when *x>500*, the change of *y* is so little that it can be ignored. This decision is dependent on the actual situation and data size.

For $\Delta x=1$, the recursive updating formula for *y* is

$$y_2 = \frac{e^{0.01} - 1 + y_1}{e^{0.01}}$$

4.2 Implementation of the Informal Process Discoverer Service

Figure 4.5 shows implemented class structure of the concepts described in Section 4.1 and in Chapter 3. As shown in the center, the main class is implemented based on the Spring Framework. The Spring Framework is an open-source Java application framework that provides a *inversion of control(IoC, also known as dependency injection (DI))* container. *Inversion of control* is a design that inverts traditional sequential control flow. Custom-written part of the program receive the flow of control from reusable code to achieve custom-defined tasks [Bur12]. *Inversion of control* allows objects to define their dependencies on other objects by defining constructor arguments. The container injects the dependencies into the main object when it is created. With this design, the modularity and scalability of the program code is guaranteed.

The field `context` represents a Spring container which is responsible for object instantiation. Here, we instruct the container on which objects to instantiate on the unit of packages. As shown in Section 4.1.3, for each phase in P_2 - P_5 in Fig. 4.2, we implemented an interface. In the recommender, at each phase, the container is simply instructed to scan the corresponding package in which the implementation of the interface is located, and the method of each implementation will be called at execution time. This way, an implementation can be easily added, deleted or modified without effecting the main interface.

In the figure, on top of `InformalProcessRecommenderSpringBased`, there are two analyzers, which bring out executions of P_2. `HumanResourceAnalyzer` analyzes one given human actor at a time. It is responsible of collecting interactions from certain user-defined Git repositories (`addtionalGitInstances`) and store the interactions related to the human actor in the interaction list. These interactions, along with those returned

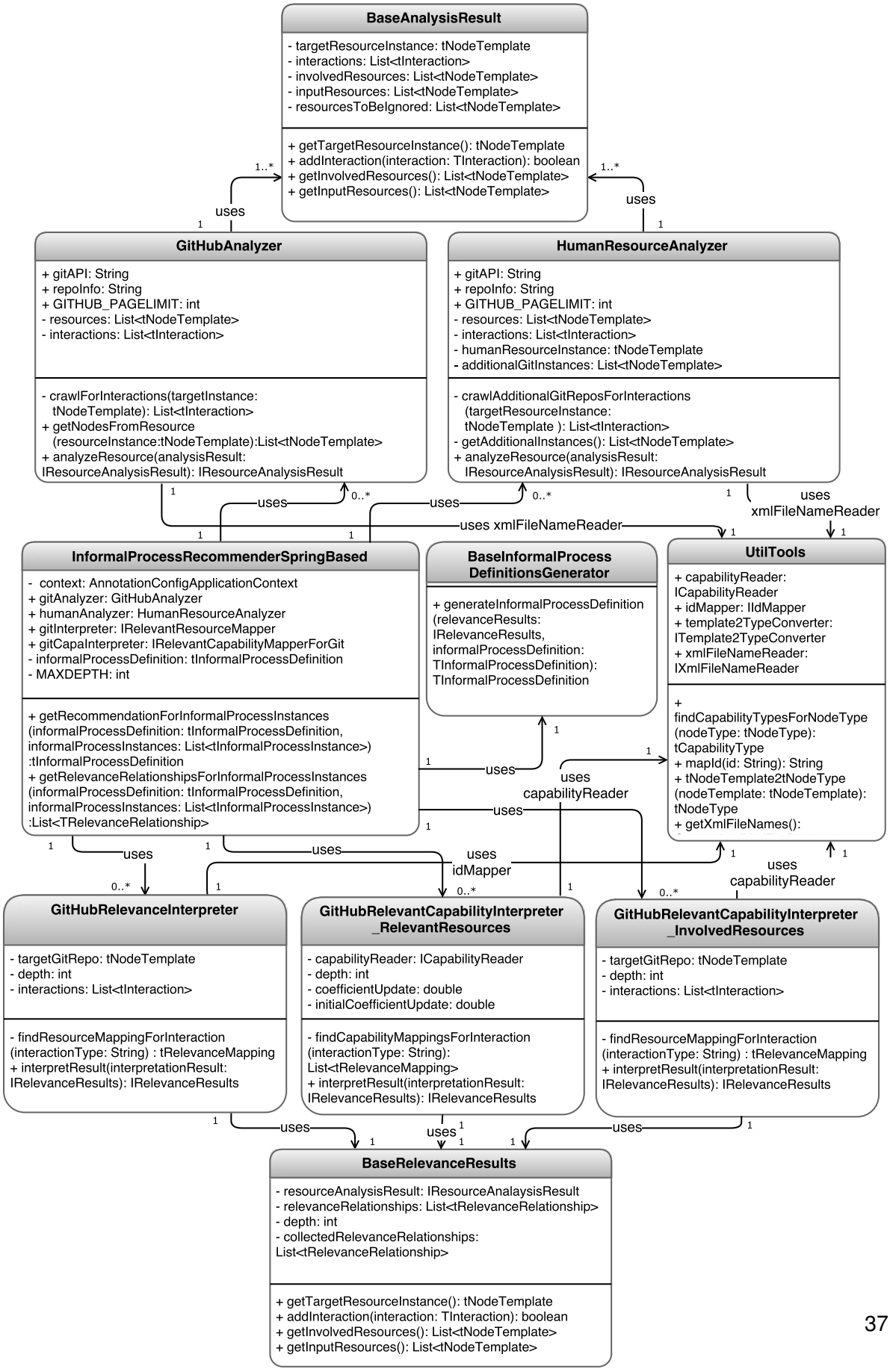

Figure 4.5: Class definition of the Informal Process Discoverer Service

by `GitHubAnalyzer`, will be stored in a `BaseAnalysisResult` object and returned to the recommender.

In P_3, we extract relevance relationships of resources out of interactions in `GitHubRelevanceInterpreter`. These relevance relationships are stored in `BaseRelevance-Results` and returned to the recommender. The recommender then starts P_4 by calling `GitHubRelevantCapabilityInterpreter_RelevantResources`, which iterates through `BaseRelevanceResults.relevanceRelationships` and updates this field by creating new relevance relationships of relevance capabilities found through capability mapping, and `GitHubRelevantCapabilityInterpreter_InvolvedResources`, which iterates through `BaseRelevanceResults.resourceAnalysisResult.involvedResources` and updates the `relevance relationships` field by looking up in a database file for predefined capabilities for each involved resource.

In P_5, a `tInformalProcessDefintion` object is "assembled" using the relevance relationships in the final returned `BaseRelevanceResult` object.

In the implementation of informal process recommender, a level of iterative analysis is included. The main idea is that before building the recommendation, P_2-P_4 are executed iteratively to attain as much information as possible. The `MAXDEPTH` field defines the max iteration depth. In each iteration, we take relevant resources of the last iteration as involved resources, and start collecting interactions based on these. To prevent the interpreters from interpreting a certain interaction twice in different iterations, we record in the field `resourcesToBeIgnored` in `BaseAnalysisResult` the involved resources of all previous iterations. At interpretation phase, since the capability correlation coefficient is supposed to evolve recursively, we retain the already-collected relevance relationships in the field `collectedRelevanceRelationships` in `BaseRelevanceResults`. In each iteration of process discovery, the update of correlation coefficient will be weaker than last iteration, since the convergence speed of the exponential formula decreases by the increase of depth d (see Section 4.1.5).

4.3 Results of Case Study

In this section, we set up different test cases and discuss the relationship between results and their corresponding test cases[1]. Here, due to rate limiting of the Git API[2], a iteration

[1]For more implementation details, please see GitLab repository https://gitlab.com/timur87/informal-process-recommender

[2]https://developer.github.com/v3/activity/events/

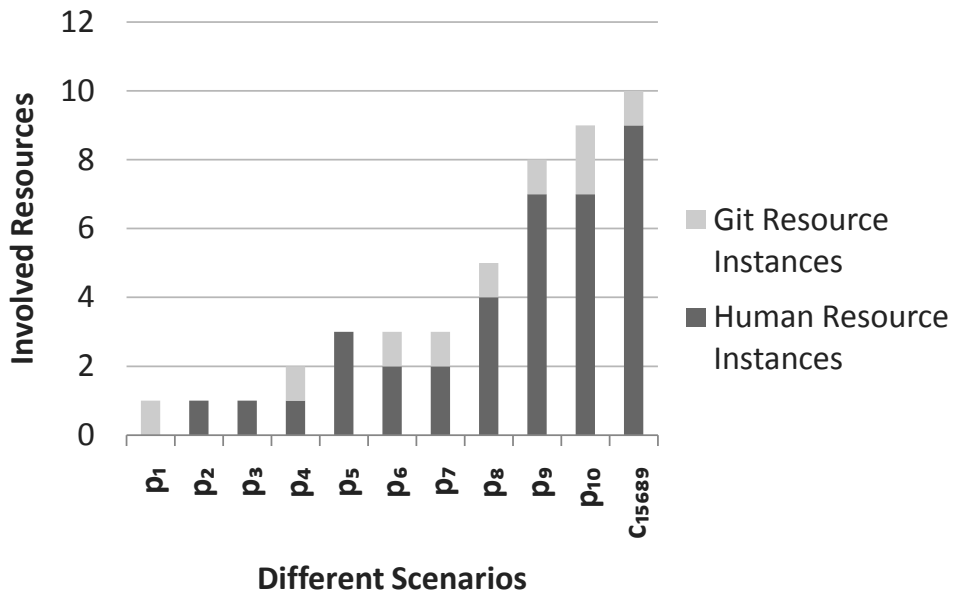

Figure 4.6: Number of involved resources in different scenarios

depth of 2 is designed. We give the results of the two iterations separately, so that the readers can see the effects of second iteration on completing the system.

Overall, 11 different scenarios are designed with varying numbers of involved resources. There are two types of involved resources: human actor and Git repository. Combinations of different numbers of these two are attempted, with also different instances and the same number. As shown in Fig. 4.6, the number of involved resources vary from 1 to 10. In the cases where the number is 1, there can either be one Git instance or a human actor instance. Theoretically, these are incomplete informal process models, with which we test the system's response to relatively extreme user inputs. As shown in the figure, in both p_2 and p_3 there are only one human actor as involved resource. Here we test the result's dependence on the choice of input involved resources. The input reveals the user's understanding on the project. In an extreme situation, if the input involved resource is an irrelevant person who did not make any interaction with the given additional Git repositories in reference, the recommender will return an empty informal process model.

In the last scenario c_{15689}, we give a list of informal process instances as input, which is a combination of the scenarios p_1, p_5, p_6, p_8 and p_9. This scenario is designed to test the system's capability of aggregating resources from several process instances and recognizing possible resource overlap.

4.3.1 Results of First Iteration

In this section, we discuss the results of the first iteration in the Informal Process Discoverer service. Firstly, the number of interactions and relevant resources are discussed. After that, the number of relevant capabilities will be shown.

4.3.1.1 Interactions and Relevant Resources

As a interval result of the two-level analysis system, the number of interactions is a symbol of the amount of information that we can finally obtain from the informal process instance(s). Thus, we discuss the relationship of interactions on input involved resources. Of course, as a direct result, the number of relevant resources is also analyzed here.

As shown in Fig. 4.7, in general, the number of interactions tends to increase with the number of involved resources, which corresponds with our intuition: the more resources we are aware of, the more interactions/information of the process we can obtain. There are, however, certain protrudes and concaves of the curve. This phenomena leads our attention to the fact that the results of the system not only depends on the number of input resources, but also, and much more crucially, on the choice of input resources. Of course, one active human actor, who makes interactions with other resources daily, could be more effective on the analysis than multiple external contributors who only contribute or star the repository several times in the year. If a completely irrelevant resource is selected as input, the recommendation will be an empty one. This is of course a theoretically extreme situation, but provides guidance at execution phase for the user. As Fig. 4.7 shows, the curve sinks at p_3, in which we have 1 human actor as involved resource-as many as in p_2. The difference is that the human actor in p_2 is a relatively more active one. As a result, the number of relevant resources at p_3 also shows a depression, only the decreasing extent being much smaller than the number of interactions. This is due to the fact that one relevant resource may make multiple interactions. In the best case (p_{10}), we found 56 relevant resources, including 48 human actors and 8 Git repositories. The results are recorded on the online spread sheet[3], in which detailed information about each scenario is given. Please note that the results may change when executed at different time, since events in the Git API only contain information of the last 90 days.

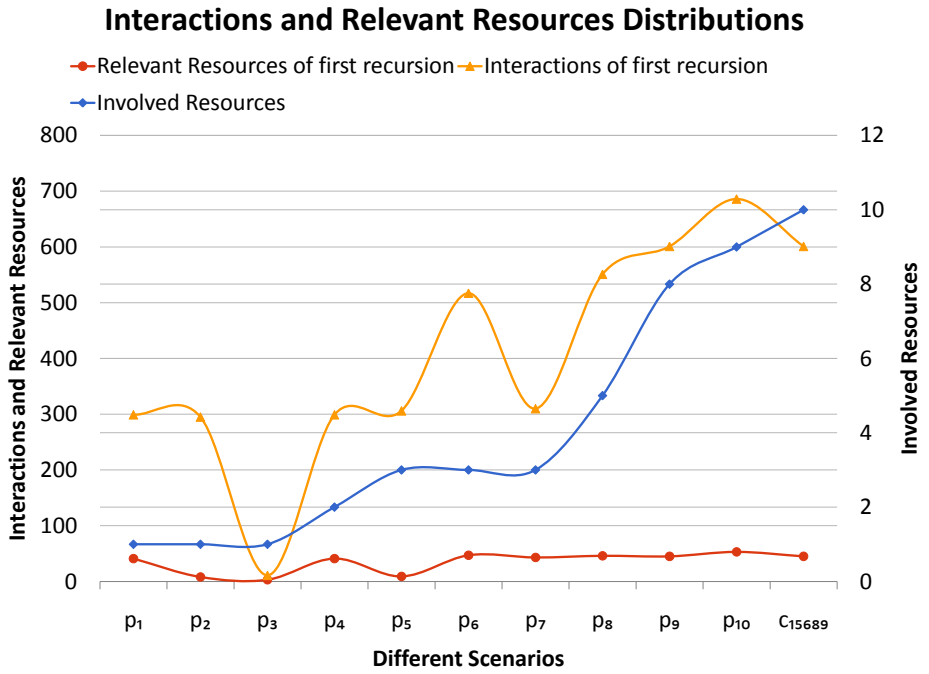

Figure 4.7: Number of interactions and relevant resources of first iteration

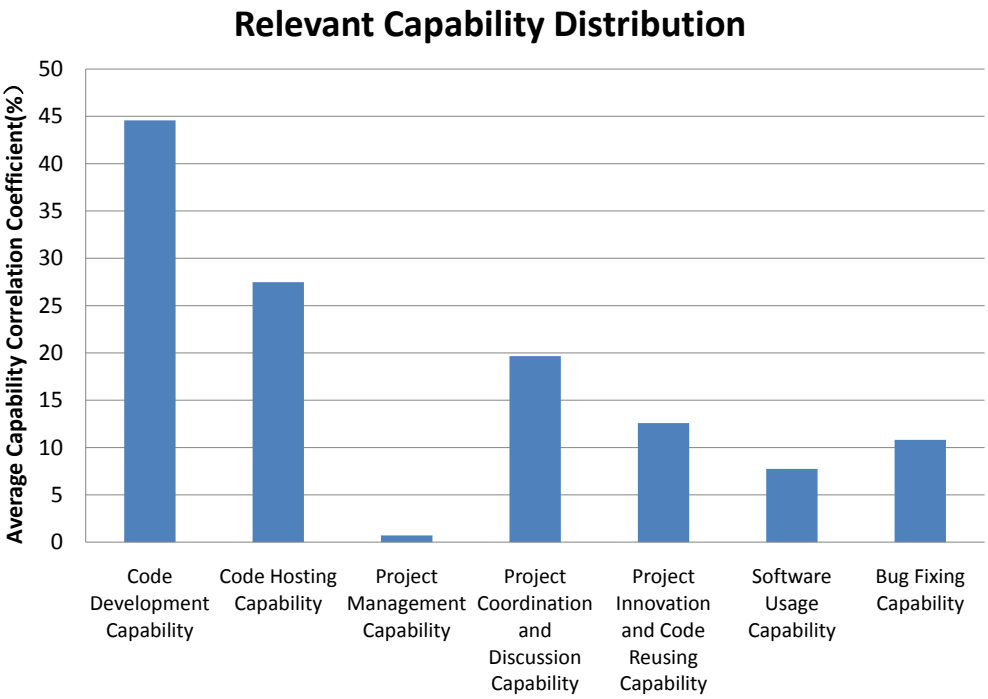

Figure 4.8: Distribution of Relevant Capabilities in First Iteration

4.3.1.2 Relevant Capabilities

The distribution of relevant capabilities is shown in Fig. 4.8. As the figure indicates, there are in total 7 types of capabilities defined, each one corresponds to a set of interactions, where different sets of interactions of capabilities may be overlapping. Here, Code Development Capability is interpreted to have the most correlation coefficient with almost 0.45. This results from a relatively large value of weight of interactions corresponding to the particular capability specified in the test file. Following it is Code Hosting Capability which is exposed by human interactions with GitHub repositories. The percentage of this capability stands in second place because of the large value of interactions interpreted. Besides that, Project Coordination and Discussion Capability is also highly relevant. On contrast, the average correlation coefficient value of Project Management Capability is less than 1 percent. This is because the interaction types corresponding to this capability type such as creating, deleting, membering and publicing a repository is either not frequently executed or overwritten by interactions that occurred later in time. It shows that the jclouds project is more-or-less an autonomous one.

4.3.2 Results of Second Iteration

In this section, we discuss the results of the second iteration in the IPD service. Here, the number of interactions, relevant resources and correlation coefficients of relevant capabilities are discussed comparing to these results of the first iteration.

4.3.2.1 Number of Interactions in the Second Iteration

The number of interactions analyzed in the second iteration are represented in this section in compare with results of first iteration in Section 4.3.1.1. In Fig. 4.9, numbers of interactions found in first and second iteration are shown respectively with the two lower curves. Interestingly, in scenarios p_1-p_4, the number of interactions in both iterations seem to expose similar tends. From p_5 on, they seem to be in some extent compensatory–if we have a relatively lower amount of interactions in the first iteration, the second iteration will find more interactions, as if there was some kind of negative feedback. In fact, if we view these two curves as if the number of interactions were a function of the different scenarios, from p_5 to c_{15689}, these two curves are nearly symmetric to the (virtual) constant function $y=425$. Also, when we look at the sum

[3] https://goo.gl/Fo6ZZF

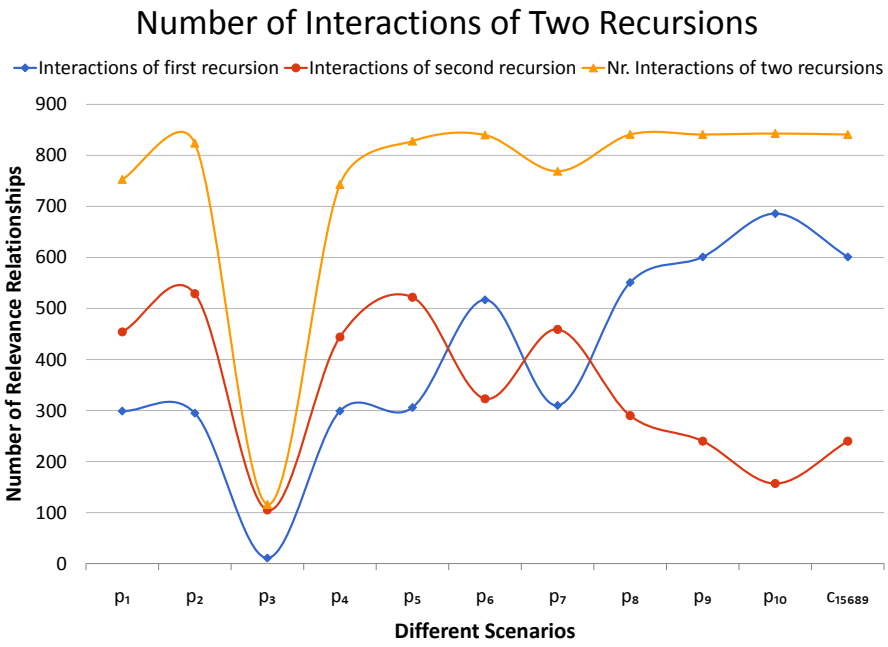

Figure 4.9: Number of Interactions in Two Iterations

of interactions found in two iterations, it stays surprisingly constant except in the case p_3.

In fact, this corresponds to what is expected in the tests and demonstrates the necessity of the second iteration. In reality, the system user could have much or little information about the informal process he is to analyze. In scenarios p_1-p_5, the input information of the system is relatively incomplete. For example, in p_2 and p_3, we assume that the process expert only knows one human performer in the process; in p_1, the expert is aware of no humans but only one GitHub repository in the process. We assume that in a same process at a point of time, the number of historical interactions is constant. Since the tests are launched within a small time window, the process state could be seen as constant. In these relatively incomplete situations, the system still needs a way to find as much information as possible. As the blue curve shows (the one that starts at the lowest point), the first iteration returns a small number of interactions, compared to the cases p_8-c_{15689}, where the input is more complete. To make up for this, the second iteration takes advantage of the results of the first one. In p_1, the second iteration crawls the human actors found in the first one using background Git repositories and returns 454 interactions. In p_2, the second iteration crawls the Git repositories interpreted in the first iteration, and returns 529 interactions. This provides a guarantee for us to collect sufficient information about resource interactions so that important resources in the process are not neglected. An exception is p_3, where the number of interactions

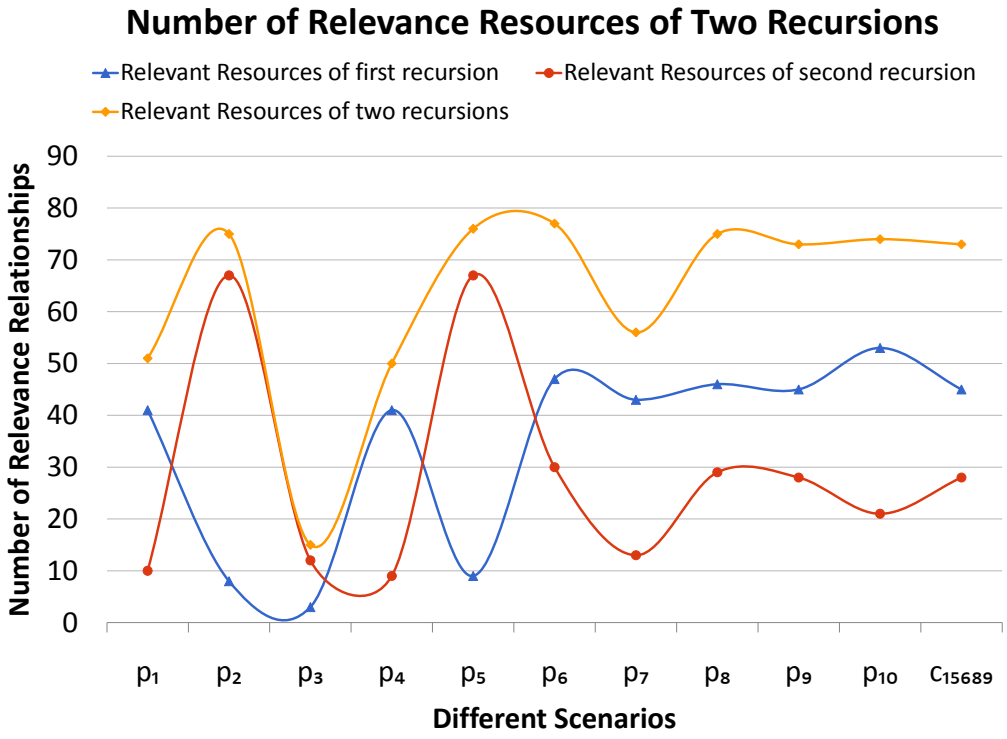

Figure 4.10: Number of Relevant Resource in Two Iterations

in two iterations is less than one eighth of the other scenarios. The reason is that in this scenario, a less active person is chosen. According to the records of GitHub events API, this person only made interactions to three of the background Git repositories. As a result, only these three are crawled in the second iteration, resulting in an insufficient sum of interactions. This indicates that by including such a resource in the input, a bias is introduced in the analysis and the process will not be understood in the expected way.

4.3.2.2 Number of Relevant Resources in the Second Iteration

In theory, the amount of information of a process, no matter how it is measured, stays constant in a small time period. Since the tests of different scenarios are launched in the same day in a time window of 3 hours, this could be assumed true. Suppose that in an informal process at a point of analysis time, the number of relevant resources stays constant, i.e. external contributors who participate in the process and IT-resources which are utilized during analysis time can be neglected. At this point, we are concerned about finding a sufficient number of relevant resources. Based on the results of p_8-c_{15689}, we

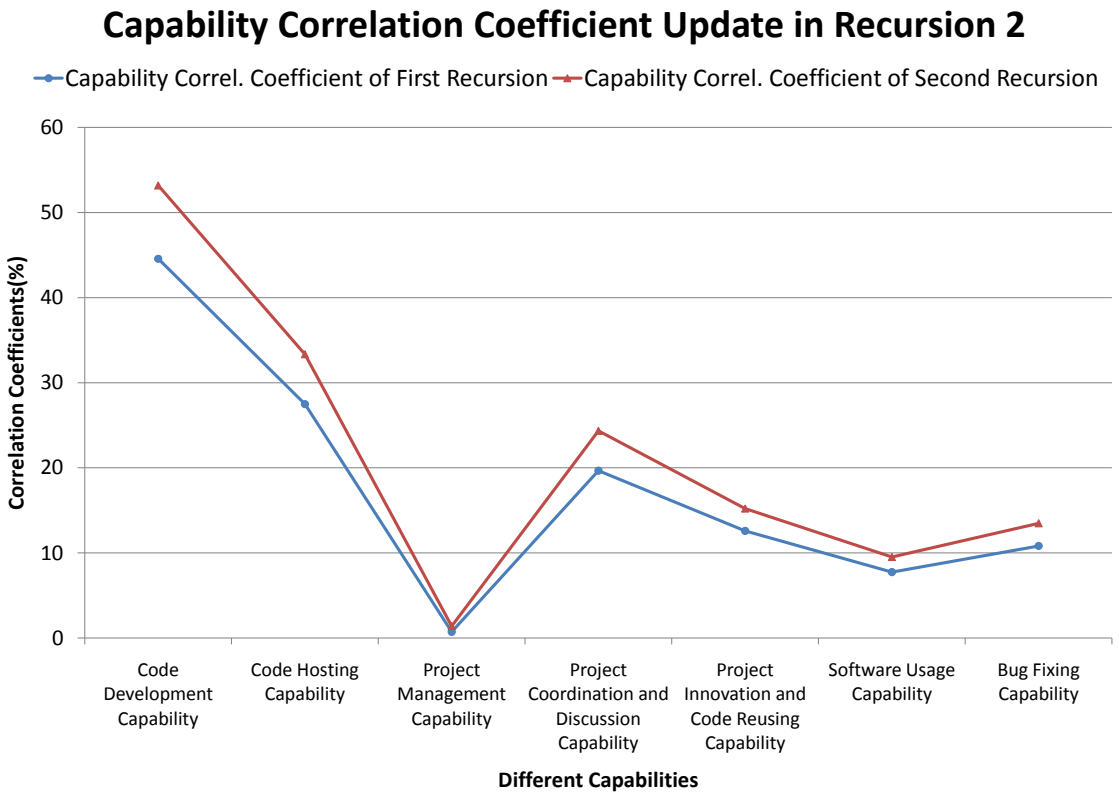

Figure 4.11: Value of Correlation Coefficient of Relevant Capabilities in Two Iterations

can assume that the results are converging at second iteration. The term "convergence" here means that our results are approaching the actual set of results.

In scenarios which found less relevance resources, such as p_1, p_3 and p_7, further iterations may be needed. Due to time limits, iterations with depth more than 2 are not implemented in this work. However, one can deduce with existing results that even with bad inputs such as p_3, it is highly possible that multiple iterations bring satisfactory result. As introduced in Section 4.3.2.1, the second iteration of p_3 returns a set of humans interpreted from the interactions to Git repositories found in iteration one. In a possible third iteration, again, we can follow these persons to their contributed Git repositories. And in fourth iteration, the Git repositories are crawled to find more relevant human actors...In a word, the more iterations launched, the bigger the possibility will be that we trace the real relevant resources, no matter how irrelevant the inputs are.

4.3.2.3 Capability Correlation Coefficients in Second Iteration

The Fig. 4.11 shows an increase in capability correlation coefficients in the second iteration. On average, the second iteration brings an increase of 22.16% of correlation coefficients. Due to our design of the iterative correlation coefficient formula, the values of correlation coefficients will increase more slowly when iteration depth increases, and will never be larger than 1.0. Moreover, the final distribution of correlation coefficient of different capability types stays similar as that of the first iteration.

4.3.3 Execution Time Distribution

Since the recommender system uses the GitHub API to retrieve information from the GitHub server, the execution time of the system depends greatly on the internet speed. As a result, the execution time diagram is only given here in order to provide intuitions for the reader about the system execution, the readers should bare in mind that given different internet conditions, the data may vary largely.

As shown in Fig. 4.12, the most time-consuming task in the analysis process is the analysis of resources(see red and purple part of the columns). The analysis of resources in second iteration takes as long as 50373 milliseconds. For the whole execution process, the average execution time is 1020223 milliseconds, i.e. 17 minutes. In fact, since time complexity is not our first concern in implementing the system, naive methods are mostly used to realize resource analysis. To shorten execution time, buffers can be set up to store interactions so that a certain repository would not be crawled several times in order to analyze different human actors.

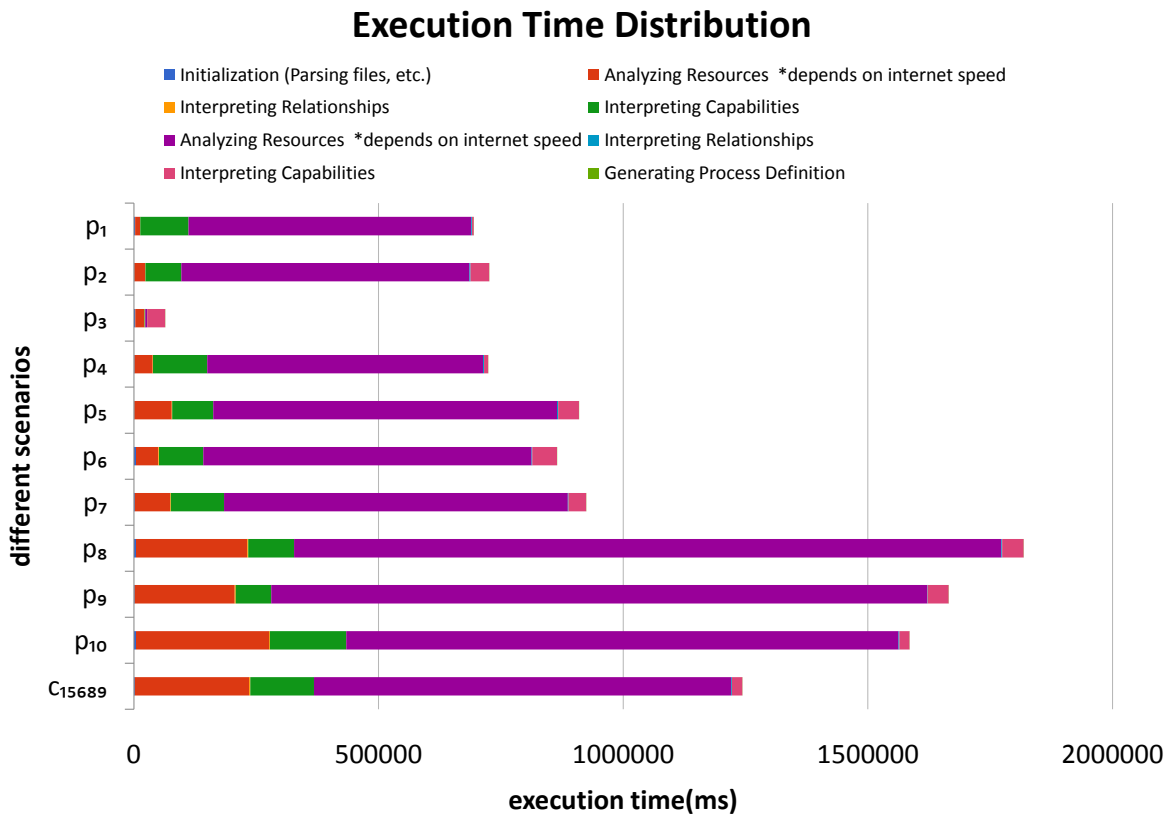

Figure 4.12: Distribution of Execution Time on Different Steps of System

5 Related Work

Extensive researches have been launched on process recommending systems to tackle with the lack of flexibility in ad-hoc processes. These systems often require that users follow an initial predefined business logic, but allow them to deviate from this business logic during process runtime in predefined ways [BDWL12]. For example, Sadiq et al. [SSO01] present a workflow model that includes only the partial process definition by categorizing process workflow fragments into sequences, forks, synchronizations, etc., and defining possible change-of-flow patterns for each fragment. Moreover, Adams et al. [PW08] provide a state-transition approach by providing a set of predefined activities and making run-time choices of activities to be executed according to the specific process state. These approaches still make rigid assumptions about process activities and require that the workflow-based process model foresees all possible variations, which is generally unrealistic for process modeling and execution [BDWL12]. Apart from these approaches, there exist workflow-mining techniques such as [DHA06], in which Dustdar et al. proposed a event log mining tool based on the process-aware collaboration system Caramba [Dus04] which converts Caramba logs into control-flow structures. In addition, Agrawal et al. [AGL98] present a workflow-logs mining approach in discovering an activity graph. However, it is not addressed how recommendations could be generated based on the results.

The above approaches mainly focus on detecting possible process workflows and give dynamic recommendations of process activities at run-time. Apart from these, more similar as in our work, works have been done to solve the problem of "finding expertise", e.g. finding skilled human resources for a certain topic. By crawling code repositories and employee databases, Begel et al. [BKZ10] generate a resource graph in which nodes represent different resources such as persons and file folders, and edges indicate relationships such as "mentions" or "modifies". This work does not fall in the context of business processes, but rather supporting collaborations. However, similar as in our work, besides human resources, IT resources such as code reporitories are also considered in the work. In fact, the discovery of relevant resources is also supported. Recommendations of human experts are given by detecting the reachability between a human resource and the IT resource associated with the given topic. One difference between this work and our work may be that this work does not contain the concept of relevant capabilities and does not address the degree of relevance. Another similar work

of expertise finding is [MH02], which proposed a method in deciding "level of expertise" by crawling code in version control systems and counting the number of lines of changes made by an expert. Based on these results, a Web-based tool visualizing an ordered list of people exposing a certain type of experience is given. Similarly, Balog et al. [BAR06] propose a strategy to find experts for a certain topic by locating the documents on topic, and then associating these documents with experts using named entry recognition.

In this work, we use a correlation coefficient for resources to depict their level of relevance in the process. We used a method to calculate resource relevance levels. In other works, this has been addressed in other dimensions, such as team connectivity. Sellami et al. present a method of converting syntactic logs, such as XES [VBDA10], into semantic logs for socio-space modeling and discovery [SGD13]. In a more detailed way, Dorn et al. propose a weighted social network graph in which nodes represent humans, and edges are weighted proportional to the frequency of interactions between two nodes [DSSD11]. In Dorn's work, a set of human performers are given as process recommendation which maximizes the experience level for every type of skill and minimizes the value of the team distance function. In our work, similar aims have been reached by giving a set of relevance relationships with correlation coefficients indicating both capability levels and resource relevance.

6 Conclusion and Future Work

In this chapter, we will provide a conclusion on this thesis work and possible future directions.

6.1 Conclusion

Traditional process modeling approaches focus on highly structured processes which have strict predefined business logic. However, there exist many informal processes, in which decisions on made at process runtime based on human experience. Thus, these processes could not be modeled using a sequence of predefined activities. To tackle this issue, the Informal Process Essentials modeling approach is proposed. It is a resource-centric approach which retains best practice by describing resources needed in informal processes. Based on this approach, the InProXec method for informal process initialization and discovery on an automated basis is introduced. In this thesis work, a validation of the concepts included in the InProXec method is given. We implemented the discovery of IPE models in the Informal Process Discoverer (IPD) service based on a mapping mechanism. Using the IPD service, a case study on the jclouds project is launched. The results of the case study have shown that by interpreting the interactions enacted during the informal process, relevant resources and relevant capabilities can be discovered. Moreover, the relevant capability levels can be calculated through analysis of the interactions using an evolving exponential function introduced in this work. As a result, an informal process model containing these relevant resource and capabilities can be constructed and given to business experts as recommendations.

6.2 Future Work

In this approach, the focus of implementing the system has been put on the validation of concepts in the InProXec method. Here, optimization of the analyzing process in terms of time complexity has not been an emphasis. For example, for different human actors, the event logs of background GitHub repositories might be crawled repeatedly,

i.e. the GitHub server is requested for the same set of data multiple times. This results in a optimizable service response time (17 minutes) of the service. To tackle this issue, efforts can be made to cache the already crawled Git interactions and reuse them when another human actor is to be analyzed.

Since the service is made to be extensible by defining an overall mapper interface, other analyzers can be added by extending the interface. For example, an analyzer for the cWiki service could be added in order to discover knowledge resources such as process documentations. Furthermore, email archives could also be mined and interpreted for relevant resources and capabilities, which would require understanding of natural language on some extent.

A List of Acronyms

The following list contains all the acronyms used in this document.

BPEL Business Process Execution Language

BPMN Business Process Model and Notation

BPM Business Process Management

IPE Informal Process Essentials

IPD Informal Process Discoverer

XML eXtensible Markup Language

API Application Programming Interface

TOSCA Topology and Orchestration Specification for Cloud Specifications

IT Information Technology

Bibliography

[AG07] W. M. P. van der Aalst, C. W. Günther. “Finding Structure in Unstructured Processes: The Case for Process Mining.” In: *ACSD*. Ed. by T. Basten, G. Juhás, S. K. Shukla. IEEE Computer Society, 2007 (cit. on p. 13).

[AGL98] R. Agrawal, D. Gunopulos, F. Leymann. “Mining Process Models from Workflow Logs.” In: *EDBT*. Ed. by H. J. Schek, F. Saltor, I. Ramos, G. Alonso, H. J. Schek, F. Saltor, I. Ramos, G. Alonso. Vol. 1377. Lecture Notes in Computer Science. Springer, 1998 (cit. on p. 49).

[BAR06] K. Balog, L. Azzopardi, M. de Rijke. “Formal Models for Expert Finding in Enterprise Corpora.” In: *Proceedings of the 29 Annual ACM Conference on Research and Development in Information Retrieval (SIGIR 2006)*. 2006 (cit. on p. 50).

[BBLS12] T. Binz, G. Breiter, F. Leymann, T. Spatzier. “Portable Cloud Services Using TOSCA.” English. In: *IEEE Internet Computing* 16.03 (May 2012) (cit. on p. 25).

[BDWL12] T. Burkhart, C. Dorn, D. Werth, P. Loos. “A Flexible Approach Towards Self-Adapting Process Recommendations.” In: *COMPUTING AND INFORMATICS* 30.1 (2012) (cit. on p. 49).

[BE02] T. R. Browning, S. D. Eppinger. “Modeling impacts of process architecture on cost and schedule risk in product development.” In: *IEEE Trans. Engineering Management* 49.4 (2002) (cit. on p. 11).

[BGD+07] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, G. Hsu. “Open Borders? Immigration in Open Source Projects.” In: *Proceedings of the Fourth International Workshop on Mining Software Repositories*. MSR ’07. Washington, DC, USA: IEEE Computer Society, 2007 (cit. on p. 15).

[BKL14] T. B. U. Breitenbücher, O. Kopp, F. Leymann. “TOSCA: Portable Automated Deployment and Management ofCloud Applications.” In: *Advanced Web Services*. Ed. by T. B. U. Breitenbücher, O. Kopp, F. Leymann. New York: Springer, Jan. 2014. Chap. TOSCA: Portable Automated Deployment and Management ofCloud Applications, pp. 527–549 (cit. on p. 25).

[BKZ10] A. Begel, Y. P. Khoo, T. Zimmermann. “Codebook: discovering and exploiting relationships in software repositories.” In: *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*. ICSE ’10. Cape Town, South Africa: ACM, 2010 (cit. on p. 49).

[Bur12] K. Burns. “Beginning Windows 8 Application Development: XAML Edition.” In: Berkeley, CA: Apress, 2012. Chap. Inversion of Control, pp. 165–174 (cit. on p. 36).

[DHA06] S. Dustdar, T. Hoffmann, W. M. P. van der Aalst. “Mining of ad-hoc business processes with TeamLog.” In: *Data Knowl. Eng.* 55.2 (Jan. 6, 2006) (cit. on pp. 11, 49).

[DM03] T. Dufresne, J. Martin. *Process Modeling for E-Business*. INFS 770 Methods for Information Systems Engineering: Knowledge Management and E-Business. 2003 (cit. on p. 11).

[DSSD11] C. Dorn, F. Skopik, D. Schall, S. Dustdar. “Interaction mining and skill-dependent recommendations for multi-objective team composition.” In: *Data Knowl. Eng.* 70.10 (2011) (cit. on p. 50).

[Dus04] S. Dustdar. “Caramba: A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams.” In: *Distributed and Parallel Databases* 15 (2004) (cit. on p. 49).

[Gar16] Gartner. *Business process management (BPM)*. http://www.gartner.com/it-glossary/business-process-management-bpm/. 2016 (cit. on p. 11).

[GK02] A. Gunasekaran, B. Kobu. “Modelling and analysis of business process reengineering.” In: *International Journal of Production Research* 40.11 (2002) (cit. on p. 11).

[KBBL13] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann. “Winery - A Modeling Tool for TOSCA-Based Cloud Applications.” In: *ICSOC*. Ed. by S. Basu, C. Pautasso, L. Zhang, X. Fu. Vol. 8274. Lecture Notes in Computer Science. Springer, 2013 (cit. on p. 16).

[LR00] F. Leymann, D. Roller. *Production Workflow: Concepts and Techniques*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000 (cit. on p. 12).

[MGM+06] P. Moody, D. Gruen, M. J. Muller, J. C. Tang, T. P. Moran. “Business activity patterns: A new model for collaborative business applications.” In: *IBM Systems Journal* 45.4 (2006) (cit. on p. 11).

[MH02] A. Mockus, J. D. Herbsleb. “Expertise Browser: A Quantitative Approach to Identifying Expertise.” In: *Proceedings of the 24th International Conference on Software Engineering*. ICSE ’02. Orlando, Florida: ACM, 2002 (cit. on p. 50).

[Nur08] S. Nurcan. “A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts.” In: *IEEE Computer Society* (2008) (cit. on p. 12).

[OAS13] OASIS. *Topology and Orchestration Specification for Cloud Applications Version 1.0*. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html. [Online; accessed 07-April-2016]. 2013 (cit. on p. 25).

[OMG11] O. M. G. (OMG). *Business Process Model and Notation (BPMN) Version 2.0*. Tech. rep. Object Management Group (OMG), Jan. 2011. URL: http://taval.de/publications/BPMN20 (cit. on p. 11).

[PW08] A. Polyvyanyy, M. Weske. “Flexible Process Graph: A Prologue.” In: *OTM Conferences (1)*. Vol. 5331. Lecture Notes in Computer Science. Springer, 2008 (cit. on p. 49).

[Ros11] A. Rosenfeld. “BPM: Structured vs. Unstructured.” In: *BPTrends* (2011) (cit. on p. 11).

[SBBL14a] C. T. Sungur, T. Binz, U. Breitenbücher, F. Leymann. “Informal Process Essentials.” In: *EDOC*. IEEE Computer Society, 2014 (cit. on pp. 12, 13, 19–21, 23).

[SBBL14b] C. T. Sungur, T. Binz, U. Breitenbücher, F. Leymann. “Informal Process Essentials.” English. In: *Proceedings of the 18th IEEE Enterprise Distributed Object Conference (EDOC 2014)*. Ulm: IEEE Computer Society, Sept. 2014 (cit. on p. 12).

[SBLW15] C. T. Sungur, U. Breitenbücher, F. Leymann, J. Wettinger. “Executing Informal Processes.” Englisch. In: *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2015)*. ACM, Dezember 2015 (cit. on pp. 12, 13, 19, 22).

[SGD13] R. Sellami, W. Gaaloul, B. Defude. “Process socio space discovery based on semantic logs.” In: *Journal of internet technology (JIT)* 14.3 (2013) (cit. on p. 50).

[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. 11).

[SKL14] C. T. Sungur, O. Kopp, F. Leymann. “Supporting Informal Processes.” Englisch. In: *The 6th Central European Workshop on Services and their Composition (ZEUS 2014)*. Potsdam: CEUR-WS.org, Feb. 2014.

[SSO01] S. W. Sadiq, W. Sadiq, M. E. Orlowska. “Pockets of Flexibility in Workflow Specification.” In: *Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, Yokohama, Japan, November 27-30, 2001, Proceedings*. 2001 (cit. on p. 49).

[Sta07] O. Standard. *Web Services Business Process Execution Language Version 2.0*. Tech. rep. OASIS, 2007 (cit. on p. 11).

[VBDA10] H. Verbeek, J. Buijs, B. van Dongen, W. Aalst. "XES, XESame, and ProM 6." In: *Information Systems Evolution*. Ed. by P. Soffer, E. Proper. Vol. 72. Lecture Notes in Business Information Processing. Springer-Verlag, Berlin, 2010 (cit. on p. 50).

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007 (cit. on p. 11).

All links were last followed on April 14, 2016.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature