# Identifying Relevant Resources and Relevant Capabilities of Collaborations – A Case Study

C. Timurhan Sungur*, Uwe Breitenbücher*,
Oliver Kopp†, Frank Leymann*, Mozi Song*, Andreas Weiß*
*IAAS, †IPVS
University of Stuttgart, Germany
lastname@informatik.uni-stuttgart.de

Christoph Mayr-Dorn and Schahram Dustdar
Distributed Systems Group
Vienna University of Technology, Austria
{mayr-dorn,dustdar}@dsg.tuwien.ac.at

***Abstract*—Organizational processes involving collaborating resources, such as development processes, innovation processes, and decision-making processes, typically affect the performance of many organizations. Moreover, including required but missing, resources and capabilities of collaborations can improve the performance of corresponding processes drastically. In this work, we demonstrate the extended Informal Process Execution (InProXec) method for identifying resources and capabilities of collaborations using a case study on the Apache jclouds project.**

***Index Terms*—Organizational performance; resource discovery; capability discovery; relevant resources; relevant capabilities; informal processes; unstructured processes**

## I. Fundamentals: InProXec Method

Using the Informal Process Essentials (IPE) approach, business experts can model business processes based on their resources without necessarily specifying their activities. Thus, the IPE approach is also suitable for processes involving unpredictable activities during the modeling, i.e., *informal processes*. Process models created following the IPE approach [1] rely on autonomous agents, e.g., human resources, that are capable of driving processes to success. The approach enables a certain degree of automation by engaging the required interrelated resources towards organizational intentions automatically. Moreover, it provides support for autonomous agents by including the resources they require during the process execution such as data resources and tools. Defining informal process models starts with defining their *main intentions*. After specifying main intentions of informal processes, business experts add resource definitions describing the resources that achieve the respective main intention such as a Java developer and a Git repository. *Resource definitions* are the representations of organizational resources available in modeling and execution systems of informal processes. Business experts can specify resource definitions indirectly using capabilities. An organizational *capability* is the ability to perform a productive task repeatedly using one or more organizational resources. During informal process enactments, *resource models* containing resource definitions and their relationships can be updated dynamically during runtime by adding and removing resources to meet emerging requirements. Thus, informal process instances of the same process model contain typically different resource models. Implementing the IPE approach in organizations requires the application of the InProXec method [2] with different phases. The InProXec method enables modeling of informal processes and automated provisioning of resources modeled in these processes. The method is composed of three main phases: Integrating Resources of Informal Processes ($P_1$), Modeling Informal Processes ($P_2$), and Executing Informal Processes ($P_3$). In this work, we subsume phases Informal Process Model Compilation ($P_3$) and Model Deployment and Runtime ($P_4$) presented in our previous work [2] as Executing Informal Processes ($P_3$).

*Integrating Resources of Informal Processes ($P_1$).* The first phase aims for creating the required infrastructure to enable modeling and automated initialization of informal processes. Thus, it is conducted before modeling and initializing informal processes. An indispensable part of informal processes are resources, as they drive processes to successful outcomes. Thus, modeling tools of informal processes need to present business experts all resource definitions important for informal processes of the respective organization. To present different resources provided by different services, technical experts develop *domain managers* that convert *domain-specific resource definitions* into actual resources working towards informal process intentions. The conversion requires allocating the defined resources. Therefore, technical experts develop *execution environment integrators* capable of allocating resource definitions included in informal process models ($P_1$). After completing this phase, business experts can create informal process models ($P_2$) using the resources provided by domain managers and initialize these models automatically ($P_3$) using execution environment integrators.

*Modeling Informal Processes ($P_2$).* During this phase, business experts create informal process models using the resource definitions made available during $P_1$. Firstly, business experts create organizational intentions and organizational capabilities provided by resources. Thereafter, they create strategies for intentions. To implement strategies, business experts create informal process models. After creating models, upon the presence of initial contexts, corresponding processes are initialized as described in $P_3$.

*Executing Informal Processes ($P_3$).* During this phase, execution environments integrators built in $P_1$ initialize informal process models created in $P_2$ by allocating the resource defini-

978-1-4673-9933-3/16/$31.00 ©2016 IEEE

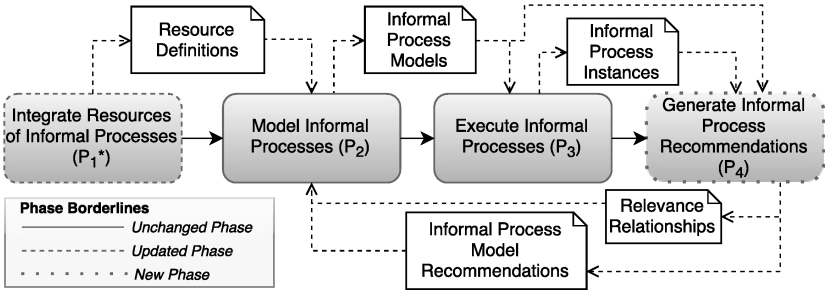


Figure 1. Extended InProXec method

tions contained in informal process models automatically. After allocating resources defined in an informal process model, execution environment integrators associate additional instance data such as instance location with resource definitions to create *resource instances*. Resource instances and resource definitions are representations of organizational resources inside of modeling and execution systems of informal processes. Resource instances represent allocated and engaged resources for specified intentions whereas resource definitions represent available organizational resources. An organizational resource defined by a resource definition may actually exist or may be firstly created during process initialization. Informal process models and instances are artifacts contained in informal process modeling and execution systems.

## II. Extended InProXec Method

As shown in Section I, we extend $P_1$ of the InProXec method and add a new phase to the end to enable discovery of relevant resources and capabilities. $P_2$ and $P_3$ stay unchanged. The extended method uses involved resources as a basis. Thus, we start by describing the term "involved resources" more precisely to support the comprehension of the extensions. As business experts update resources of informal process instances based on emerging requirements dynamically, the resource instances contained in different process instances of the same model can vary. Thus, *involved resources* of an informal process model are not only the resources captured in the respective model, but all resources contained in instances of the model. For instance, participants of a process instance can add an additional Git repository during the process execution. In this case, the additional resource is also an involved resource although it is originally not contained in the informal process model.

During the accomplishment of intentions specified in an informal process, involved resources interact with other resources. *Interactions* can occur in different environments, e.g., physical or digital, resulting in different kinds of interaction contents. Interactions provide resourceful information for finding *relevant resources* in the context of informal processes models. We consider each resource instance interacting with an involved resource directly or indirectly as *relevant*. Indirectly means that resources communicating with relevant resources are also considered relevant. For instance, consider that an involved resource edits a Wiki document, which is also updated by another person who is not involved in the respective process. In this case, the Wiki document and this second person are considered as relevant resources due to the interactions between (i) the involved resource and the Wiki resource and (ii) a relevant resource (the Wiki resource) and the second person. Neither informal process models nor informal process instances contain relevant resources, although containment will likely increase the performance of the process execution due to their participation in the process instances. The concept of *relevant capabilities* is built on top of relevant resources. A relevant capability is a capability that is possessed by an involved or a relevant resource. Moreover, a relevant capability is not contained in respective informal process instances or models. For instance, in case a developer updates a Wiki page, then a "knowledge storage" capability is a relevant capability as it is a capability of a relevant resource being the Wiki service.

*Integrating Resources of Informal Processes ($P_1^*$).* In addition to the steps described previously for $P_1$, in this phase, the following activities are conducted. Identifying relevant resources and capabilities requires analyzing interactions of organizational resources and, thus, an infrastructure capable of collecting resource interactions. During the first phase, technical experts develop services capable of (i) collecting resource interactions and (ii) deriving relevant resources and capabilities out of these interactions. Considering an informal process instance containing a Git repository, technical experts build services capable of (i) crawling interactions between contributors and this Git repository and (ii) deriving relevant human resources based on these interactions. Business experts update informal process models using relevant resources and capabilities identified manually. To automate addition of these, technical experts additionally develop services in this phase that are capable of creating new informal process models by adding relevant resources and capabilities to a source informal process model. At the end of $P_1^*$, an infrastructure capable of generating relevant resources, relevant capabilities, and informal process model recommendations is ready.

*Generate Informal Process Recommendations ($P_4$).* During $P_4$, the services developed in $P_1^*$ analyze interactions involving resource instances of informal process instances. First, services developed in $P_1^*$ collect interactions of involved resources and relevant resources. Based on analyzed interactions of involved resources, appropriate resources and

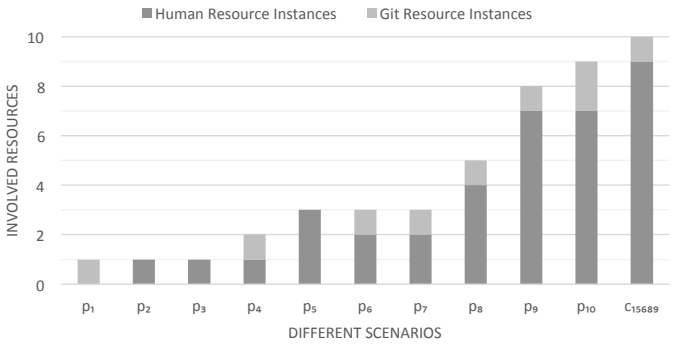

Figure 2. Scenarios used in the application

capabilities for certain jobs may be found. For example, if a certain code reviewer is identified as relevant, it is a valid recommendation to suggest this reviewer to the business expert of an informal software development process. Deciding whether recommendation fits the desired needs is up to the business expert, but nevertheless may provide a means to support them in creating new models or to give hints. The overall goal is, thus, providing a recommendation technique that suggests business experts during $P_2$ concrete resources, capabilities, and new informal process models. For example, software components developed in $P_1^*$ analyze an included Git resource for its commit messages, and, then, interpret these to create a relevant resources, relevant capabilities, and informal process model recommendations. As a result of $P_4$, (i) relationships containing relevant resources and relevant capabilities, i.e., relevance relationships, or (ii) an informal process model recommendation based on an initial informal process model and relevance relationships are presented to the business expert.

## III. Case Study on the Apache jclouds Project

In this section, we present the application of the extended InProXec method to the Apache jclouds project. Initially, we developed a core system which is capable of executing $P_4$ using the services developed in $P_1^*$. We provide the implementation of the whole case study in a GitLab repository[1]. In our case study, we focus on Git projects and human resources as these are the most relevant resources at first sight. Thus, possible involved resources of informal process models are human resources and resources representing Git repositories. The interactions serving for the respective organizational intentions are code commits, pull requests, pull request reviews, watches, and issues. To collect these interactions using the GitHub API[2], we developed one service supporting human resource instances and one supporting Git resource instances. We collected interactions of these resource instances using the Events API[3] of GitHub, which delivers events of the last 90 days. The services collecting interactions of human resources uses 17 repositories found in the jclouds organization (https://github.com/jclouds) as interaction mediums to collect interaction of involved resources. After developing services collecting interactions, we created services for identifying relevant human resources, Git resources, and a set of predefined relevant capabilities.

After conducting $P_1^*$, we validated the system with the data from Apache jclouds project. In our application, we assume the jclouds project is in $P_3$, that is, informal process models and instances already exist. Therefore, we created an informal process model targeting the resolution of a reported bug in the jclouds project and its informal process instances ($p_i$) containing different type of resource instances based on the possible execution scenarios happened in the actual project, as shown in Section II. The files containing the model and instances can be found in the GitLab repository[4]. Based on our experience, we selected 10 process instances illustrating the work. Moreover, to investigate the effects of analyzing multiple process instances at once, we use the process instances $p_1$, $p_5$, $p_6$, $p_8$, and $p_9$ at the same time during the scenario $c_{15689}$. To design the list of involved resources in different scenarios, we used the listed members of the organization (https://github.com/orgs/jclouds/people). The average execution time of $P_4$ for the scenarios presented in Section II is 17 minutes. Detailed results collected are presented in an online spreadsheet[5]. This spreadsheet provides an overview of the results such as collected relevant resources and capabilities, the number of interactions, execution times, and involved resources. These results have been calculated using the algorithms provided in the GitLab repository with the input data spanning 90 days and collected on 2016-04-01. After creating the the process instances based on the project data, we executed $P_4$ and in the following we describe the results of these. $p_1$ presents the scenario including a single Git repository, i.e., the jclouds repository (https://github.com/jclouds/jclouds). According the results of $p_1$, the most relevant Git resource and human resource are jclouds-labs and nacx (https://github.com/nacx) with the degrees of relevance 0.272 and 0.208, respectively. In the following, we refer to human resources using their GitHub user names and Git resources by their repository names. The scenarios $p_2$, $p_3$, and $p_5$ contain human resource instances with different numbers only. In $p_2$ and $p_5$, jclouds-labs was the most relevant resource with slightly different degrees of relevance, i.e., 0.508 and 0.490, respectively. Moreover, in scenario $p_3$, jclouds-site was the most relevant resource, because, in $p_3$, the involved human resource demobox was added, who was more active in this repository. In $p_6$, $p_7$, $p_8$, and $p_9$, we have the jclouds repository with different number of human resource instances. Consequently, both the number of relevant resources and the order of their relevance change. For instance, jclouds-labs is the most relevant resource for $p_6$, $p_8$, and $p_9$. Moreover, the repository jclouds-site is the most relevant resource for $p_7$ with the degrees of relevance

[1]https://gitlab.com/timur87/informal-process-recommender/
[2]https://developer.github.com/v3/
[3]https://developer.github.com/v3/activity/events/
[4]https://gitlab.com/timur87/informal-process-recommender/tree/master/src/test/resources
[5]https://docs.google.com/spreadsheets/d/1myLLwFxWIDrDGwwUSmjTHP6lpbrHAG6t-QDxEEvj58k/

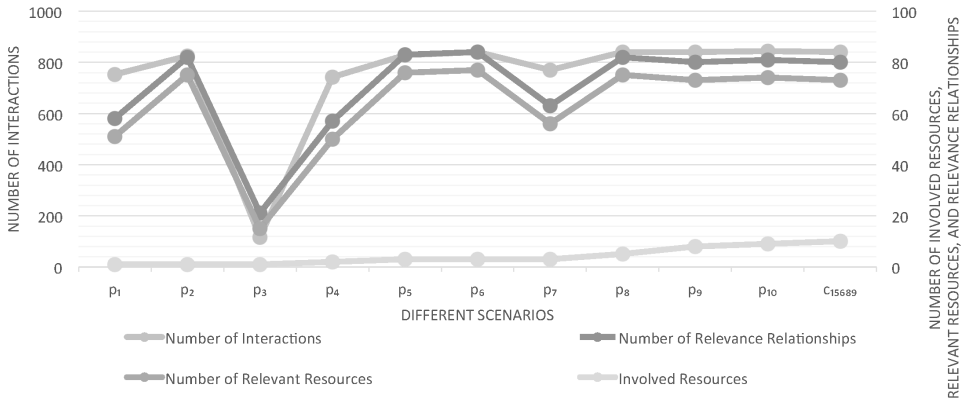


Figure 3. Relevant resources found and their relation with involved resources and interactions

of 0.636. The change from jclouds-labs ($p_6$) to jclouds-site ($p_7$) is a result of changing nacx with demobox. Replacing nacx in scneario $p_6$ with demobox in scenario $p_7$ increases the relevance of jclouds-site. Moreover, removing nacx from the the list of resource instances in $p_7$ makes him the most relevant human resource with the degree of relevance of 0.208. The combined scenario $c_{15689}$ also provides consistent results as $p_1$, $p_5$, $p_6$, $p_8$, and $p_9$, the most relevant resource is the jclouds-labs repository. We additionally include the jclouds-labs repository, which is the second most active repository in the organization based on the number of commits, in the informal process instance of scenario $p_{10}$. As presented in Section III, following this inclusion in $p_{10}$, we identified the highest number of interactions. The jclouds-site repository is the most relevant Git resource and devcsrj as the most relevant human resource.

The number of interactions typically increases based on the number and the type of involved resources. Moreover, a higher number of interactions typically results in a higher number of relevant resources, as shown in Section III. There are also other interesting results leading to other conclusions. For instance, scenarios $p_2$ and $p_3$ contain one human resource instance although the analysis result of $p_2$ provides a higher number of relevant resources (47 vs. 15) due to containing a more communicative resource. Moreover, although $p_4$ contains a higher number of involved resources than scenario $p_1$, the number of interactions and relevance relationships identified in $p_4$ is less than scenario $p_1$. The reason of this decrease is that in scenario $p_4$ 16 Git repositories were analyzed ($D_2$), whereas in $p_1$ 17 Git repositories. Only in scenario $p_4$, a smaller number (16) of repositories were analyzed. Thus, we conclude that the set of involved resources and the number of interaction services analyzed for collecting interactions play an important role and they are correlated with the number of resulting interactions and relevance relationships. All relevant resources are generated including their respective degrees of relevance[5]. Using a dynamically calculated threshold eliminating 90% of the least relevant resources and capabilities, we enabled automated generation of informal process model recommendations. In our implementation, we created different capabilities ($P_2$ of the InProXec method) such as a project coordination capability, a code development capability, a code hosting capability, a bug fixing capability, and a project management capability based on the interactions available. Among these, code development, code hosting, and bug fixing are the most relevant capabilities identified in different of the scenarios. Due to the limited space, we have to exclude the discussion of the relevant capabilities. For further details of the case study, we refer readers to the online spreadsheet provided[5].

The discovered resources, e.g., jclouds-labs and nacx, are key resources during the development of these libraries and certainly are required for the implementation of the respective process. Thus, the provided results prove that the method built on these concepts is valid and works as claimed. The selected types of interactions play a critical role and have a huge impact on the results presented. Each organization needs to determine the interactions and their impact carefully to have valid results at the end. For instance, changing the impact of a pull request message changes results dramatically. Moreover, the presented results vary from the top committers presented under GitHub statistics (https://github.com/jclouds/jclouds/graphs/contributors), e.g., the top committer is not nacx within last 90 days, as our implementation considers (i) multiple Git repositories and (ii) different types of interactions and is not limited to commits. **Future directions**: Firstly, we will evaluate our findings from this work through surveys. Secondly, we will detail the steps required to complete $P_1^*$ and $P_4$ successfully.

*Acknowledgments:* This work has been partially supported by Graduate School of Excellence advanced Manufacturing Engineering (GSaME)[6], German DFG within the Cluster of Excellence (EXC310/2) SimTech, SitOPT (Research Grant 610872, DFG), SmartOrchestra (01MD16001F, BMWi), and EU FP7 Smart-Society project (Research Grant 600854).

## REFERENCES

[1] C. T. Sungur, T. Binz, U. Breitenbücher, and F. Leymann, "Informal Process Essentials," in *EDOC 2014*. IEEE Computer Society, 2014.

[2] C. T. Sungur, U. Breitenbücher, F. Leymann, and J. Wettinger, "Executing informal processes," in *iiWAS '15*. ACM, 2015.

[6]http://www.gsame.uni-stuttgart.de/