Executing Informal Processes

C. Timurhan Sungur, Uwe Breitenbücher, Frank Leymann, and Johannes Wettinger

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

C. Timurhan Sungur, Uwe Breitenbücher, Frank Leymann, and Johannes Wettinger. 2015. Executing Informal Processes. In *Proceedings of* iiWAS '15, December 11-13, 2015, Brussels, Belgium. DOI: http://dx.doi.org/10.1145/2837185.2837225

BibTeX

```
@inproceedings{Sungur2015a,
  author    = {Sungur, Celal Timurhan and Breitenb\"ucher, Uwe and
Leymann, Frank and Wettinger, Johannes},
  title     = {Executing Informal Processes},
  booktitle = {The 17th International Conference on Information
Integration and Web-based Applications {\&} Services, {IIWAS} '15,
Brussels, Belgium, December 11-13, 2015},
  year      = {2015},
  publisher = {ACM}
}
```

Executing Informal Processes

C. Timurhan Sungur, Uwe Breitenbücher, Frank Leymann, and Johannes Wettinger
Institute of Architecture of Application Systems
University of Stuttgart
70569 Stuttgart, Germany
lastname@iaas.uni-stuttgart.de

ABSTRACT

Processes involving knowledge workers, such as decision-making processes, research processes, development processes, maintenance processes, etc. play a critical role for many organizations because they represent a valuable amount of the work an organization delivers. Therefore, supporting and automating such processes is vitally important for organizations. In our previous work, we have proposed a resource-centric approach called Informal Process Essentials (IPE) to support and to provide a certain degree of automation. The approach enables specifying required resources including autonomous agents of an informal process for accomplishing process goals through creating and initializing IPE models. Initializing an IPE model results in the acquirement of resources that collaboratively work towards the goals specified by the model. In this work, we provide an approach to automating the enactment of such resource-centric informal processes in two steps: (i) integrating resources of informal processes and (ii) executing informal processes. The approach we introduce enables the inclusion of different resource domains, e.g., IT resources, human resources, etc., and resource deployment environments, e.g., OpenTOSCA, Docker, etc. to model and enact informal processes. During the execution, the resources made available through the integration are acquired and engaged for goals of modeled informal processes. To validate the introduced concepts, we apply the approach to a detailed case study that realizes these two steps based on existing approaches and technologies, in particular, the OpenTOSCA ecosystem, an knowledge base, and an APIfication approach.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Workflow management*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Computer-supported cooperative work*

iiWAS '15, December 11-13, 2015, Brussels, Belgium
© 2015 ACM. ISBN 978-1-4503-3491-4/15/12...$15.00
DOI: http://dx.doi.org/10.1145/2837185.2837225

Keywords

Informal processes, agent-centered processes, human-centric processes, process execution, TOSCA, APIfication

1. INTRODUCTION

Many organizations rely on business process models to capture recurring procedures for enacting them in an automated fashion using corresponding business process execution engines. This *activity-centric modeling* of business processes enables automating reusable activity structures through business process modeling languages such as the Business Process Execution Language (BPEL)[1] and the Business Process Model and Notation (BPMN)[2]. During the enactment of these activity-centric processes, the modeled activities are executed as prescribed by the model. However, in various processes, modeling explicit control flows of activities in advance is not possible [20, 6]. For instance, in a software development process, required activities and their order of execution cannot be predicted beforehand as they depend on the requirements of the software to be developed. Thus, such kind of processes are typically not well-defined, i.e., *informal*, and human-centric. During execution, there is a set of critical resources that drive a process to a successful execution. For example, in a software development process, critical resources are the development environment, knowledge dissemination environments such as a Wiki software, and qualified humans such as developers. All these resources are key for achieving the specified process goals, i.e., developing the respective software. Unlike structured processes, human actors do not follow predefined activities during the enactment as they conduct these activities based on their expertise and experience. Approaches such as adaptive case management [12] or ad-hoc business process management [10] enable enacting such processes by defining activities on the fly and reusing created activity structures during future executions. However, in various cases, focusing on activities does not support human actors because the problems that they face are trivial or change dynamically and, consequently, required activities, too. In contrast to the unpredictable activities in *knowledge-intensive processes*, goals of informal processes are known before their enactment [6]. Reaching these goals requires not only certain human actors but also other resources that support them, e.g., IT resources, material resources, and knowledge resources. Based on these facts, we have proposed

[1]http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf
[2]http://www.omg.org/spec/BPMN/2.0/PDF/

a resource-centric modeling approach called *Informal Process Essentials* (IPE) [20]. The approach supports human actors by providing required resources for achieving informal process goals. By placing human actors with required capabilities into informal processes, it supports reproducing similar process results. Moreover, the presented approach is not only a stand-alone but also complementary to existing activity-oriented approaches such as adaptive case management [12] and ad-hoc business process management [10] by enabling specifying involved actors and resources required for different activities. In our previous work [20], we introduced a meta-model for specifying informal processes and an overview of a solution architecture addressing these concepts. However, we did not present an automated means to create instances of resource-centric informal process models that facilitate the achievement of prescribed process goals. The meta-model enables describing informal processes to preserve essential information such as their interrelated actor and resource definitions, the disseminated knowledge, goals of informal processes, and their context information. As the preserved information is critical for structured, unstructured, and semi-structured processes, i.e., processes containing both fixed and variable activities, informal process models provide support for processes with different granularities. Initializing resource-centric informal process models require acquiring and engaging interrelated resources towards goals of informal processes. Configuring, coordinating, and engaging different resources manually during initialization typically causes an overhead, e.g., different IT services and developers need to be installed and assigned at the beginning of a software development process. Although existing automation standards such as BPEL present necessary foundations to avoid such overheads through acquiring interrelated resources of informal processes in an automated fashion, complementary concepts to achieve an automated initialization are still missing.

In this paper, we tackle this issue of automating informal process initialization. We present the following contributions of our research: (i) a method for integrating resources of informal processes (Sect. 4.2), (ii) a method for initializing informal process models (Sect. 4.3), (iii) and a detailed proof-of-concept case study (Sect. 5). The first method enables integrating resources participating in informal processes into modeling and execution environments of resource-centric informal processes. The second method empowers the automated execution of informal process models involving resources integrated by the first method. Consequently, this work presents an approach to automating the enactment of resource-centric informal processes. In our case study, we describe how we realize the concepts introduced using the concept of APIfication [26], the OpenTOSCA ecosystem [2, 13, 4], and a comprehensive DevOps knowledge base [25].

2. MOTIVATING SCENARIO

As motivating scenario and running example of this work, we describe the informal process of developing a software application for collecting usage statistics of manufactured cars. During collection of usage statistics, the application exploits Internet of Things (IoT) technologies such as wireless and RFID sensors. Moreover, the project contains various sub-goals such as collecting requirements, setting the communication protocol between cars and the developed software, and assuring the quality of the developed code. Developing such an application requires the collaboration of various people with different expertise and skills, e.g., the cooperation of embedded software engineers who design and install sensors on cars and actual application developers. Embedded software developers contribute temporarily as external experts to the project. As a result, it's vitally important to store the information provided by them for the future usage. The project team, therefore, employs a Wiki called MediaWiki[3], on which project members can store the relevant information about the development. As usual in software development projects, developers build the project on a specific technology stack, e.g., by using Java, the Java Virtual Machine (JVM), and the build tool Maven. Although each developer can exploit her favorite integrated development environment (IDE), e.g., Eclipse, IntelliJ, NetBeans, Emacs, etc., as a common foundation, they need to rely on this technology setting. The development team executes an automated process to build the software. The management of the project is accomplished using a project management software called Redmine[4]. Moreover, participating human resources are members of a business-oriented social network called XING[5].

The resources required in this process include material, i.e., physical, resources, IT resources, and, more importantly, human actors who achieve process goals with the help of other resources. For example, developing the application requires an operating system that runs a JVM. As new goals emerge during process execution, the team may acquire new capabilities dynamically that are provided by other resources. For example, the project management can decide to support a specific IoT middleware for governing different events from car sensors. In that case, technical experts add a new software representing such a middleware.

In this scenario, the participating human actors work towards one *main goal* and certain *sub-goals*. Each resource has different semantic relationships with other resources based on the different types of interactions between the resources. For example, the project lead has administrative access permissions for the Redmine service, whereas a regular developer has more restricted access permissions. The IPE approach enables creating models involving the definitions of key actors such as managers and developers and definitions of supporting resources such as Redmine and MediaWiki. However, due to the lack of a method for initiating informal processes automatically, additional manual effort is needed. For instance, to initialize the software development process, the Redmine service needs to be installed, a qualified manager needs to be found, and Redmine needs to be configured for the manager. In this work, we provide an approach to automating the enactment of such *informally specified, resource-centric processes.*

3. FUNDAMENTALS

Many organizations document and implement value-adding activities and their structure in the form of business processes. Moreover, business process engines can enact the business process models automatically, after *configuring* the necessary IT infrastructure. Modeling, configuring, executing, and improving business processes are conducted as a cycle called *Business Process Management Life Cycle* [24, 9]. In this work, we distinguish between two roles in this life cycle: (i) *business*

[3]http://www.mediawiki.org/
[4]http://www.redmine.org/
[5]http://www.xing.com/

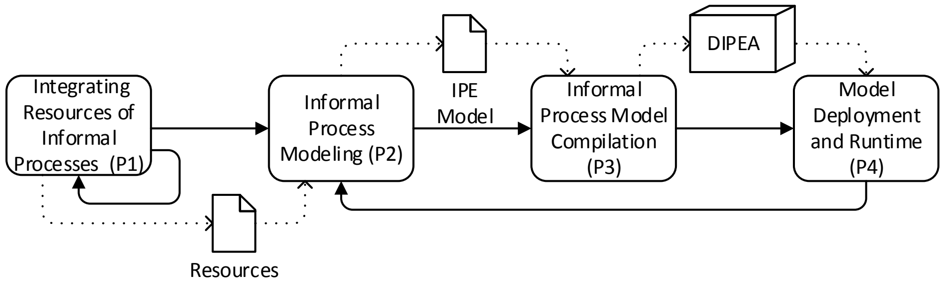

Integrating Resources of Informal Processes (P1)
Informal Process Modeling (P2)
IPE Model
Informal Process Model Compilation (P3)
DIPEA
Model Deployment and Runtime (P4)
Resources

Figure 1: Steps of the InProXec method for executing informal processes

experts that model and evaluate business processes and their execution performances, respectively, and (ii) the role of *technical experts* that configure the IT infrastructure for execution. Describing processes based on their activities is suitable for processes that are *structured* in terms of having rigid activities. However, organizations do not only involve such structured processes but also *unstructured* processes, whose exact enactment cannot or should not be prescribed in advance [6]. Different characteristics of these processes are used to name them in different scientific works. To clarify the term of informal processes, we define it as follows:

Informal processes are unstructured processes, whose execution steps cannot be modeled or are not feasible to model before their enactments. Execution steps cannot be modeled as they change during each enactment and it is not feasible as the benefit of automating the process enactment is lower than the cost of the automation. Instead, autonomous agents enact informal processes based on ad-hoc decisions, experience, and knowledge with the help of other involved resources. Thus, no formal definitions exist stating which steps in which order must be taken by which actors, but, rather, only *informal* guides and definitions of initial goals of the respective process exist.

3.1 Informal Process Essentials

In this section, we detail different concepts introduced by the Informal Process Essentials approach in our previous work [21, 20]. The approach enables the resource-centric modeling of informal processes. The concept of resources in the IPE meta-model include all kinds of resources which are valuable for achieving goals of an informal process, e.g., a developer, MediaWiki, Redmine, etc. Additionally, we distinguish between two kinds of resources regarding the time that resources are needed in informal processes: initial and on-demand resources. *Initial resources* are required at the beginning of an informal process. These resources are foreseeable during the modeling of an informal process and are believed to be enough for reaching the goals at hand. Additionally, there are also *on-demand resources* [23] that are acquired dynamically based on the emerging goals during the process enactment. A special type of resources in the IPE meta-model are *actors* that drive the process execution autonomously. The experienced human actors that already participated in certain informal processes typically take the role of business experts during modeling, as they know insights of the processes that they participated. Another specific kind of resources is *knowledge resources* that contain critical information about the process being enacted. A MediaWiki service and the documentation about a specific project are examples of knowledge resources. Human resources work on knowledge resources iteratively and they store their explicit knowledge needed for executing similar informal processes. Therefore, knowledge resources are critical for guiding human actors. Actors are typically humans; however, this role can be as well taken by certain software services that are able to drive process execution autonomously.

Actors make use of other resources to accomplish *intention* and *sub-intention* of the respective informal processes. Each intention can require certain *capabilities* that are provided by available organizational resources. A *resource organizer* is responsible for collecting resource definitions and presenting them to business experts for modeling. Moreover, each resource can be associated with other resources by *relationships*. Through relationships, business experts can model more coherent resource structures. Relationships are optional entities in each model, as they are typically used for detailing the resource structures. In this work, we address the resource models without relationships and leave the ones containing relationships as future work due to the broad context of this work.

4. INPROXEC - A METHOD TO EXECUTE INFORMAL PROCESSES

In this section, we present our *InProXec method* that follows the steps illustrated in Fig. 1. The method enables initializing informal process models in an automated fashion. In the following subsection, we present an overview of our method. Thereafter, we detail the different phases of the method.

4.1 The InProXec Method

The InProXec method consists of four different phases as shown in Fig. 1. The first phase of our method contributes to the creation of functional modeling and execution environments of resource-centric informal processes. This phase is followed by a modeling phase, in which business experts create informal process models. Hereafter, the third phase aims for producing executable informal process models. These models are then initialized in the fourth phase.

Integrating Resources of Informal Processes (P1). Many services provide information about informal process resources and means of acquiring them automatically, e.g., XING provides the information about human resources and an automated application can be used to provision a new MediaWiki service. To model informal processes using the information available about resources during modeling and to acquire modeled resources during execution, technical experts integrate such information and automation services

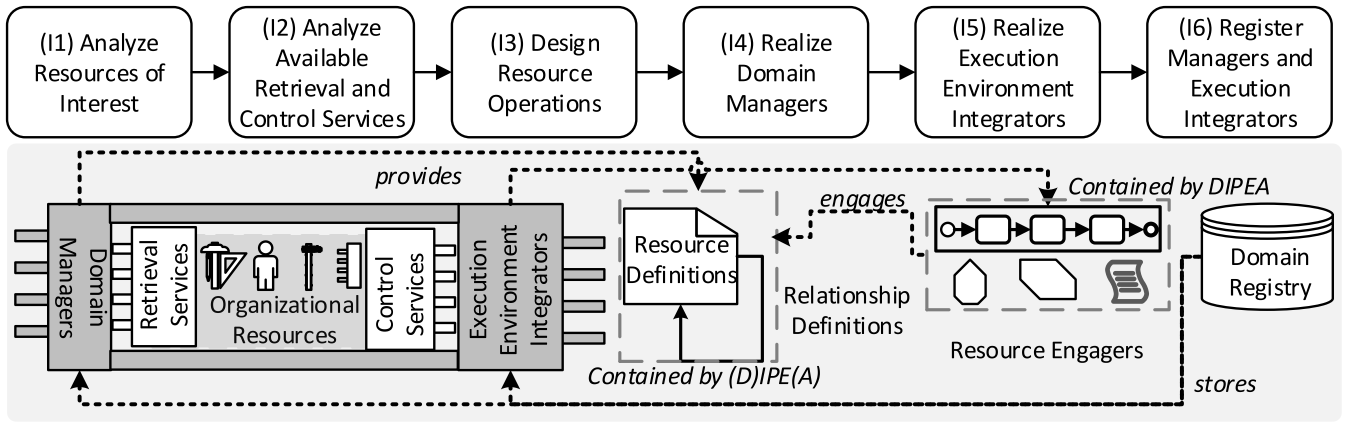

(I1) Analyze Resources of Interest
(I2) Analyze Available Retrieval and Control Services
(I3) Design Resource Operations
(I4) Realize Domain Managers
(I5) Realize Execution Environment Integrators
(I6) Register Managers and Execution Integrators
provides
Domain Managers
Retrieval Services
Organizational Resources
Control Services
Execution Environment Integrators
Resource Definitions
Contained by (D)IPE(A)
Relationship Definitions
engages
Contained by DIPEA
Resource Engagers
Domain Registry
stores

Figure 2: Integrating Resources of Informal Processes (P1)

in this phase. During the integration, they develop services (i) for retrieving information about required resources, (ii) for acquiring them, and (iii) for releasing them upon the process completion. These services are used to manage, i.e., view, acquire, and release, available organizational resources in an automated fashion. As a result, integrated resources are available in modeling environment and can be acquired / released during execution.

Informal Process Modeling (P2). In the second phase, business experts model informal processes using different IPE modeling elements, e.g., using the resources definitions made available in the first phase. Business experts create informal process models using interrelated resources aiming for a main intention containing sub-intentions. Consequently, IPE models specify informal processes and provide a means of execution in the following phases.

Informal Process Compilation (P3). IPE models only describe the goals to be achieved, the required resources, etc., but they do not provide executable functionality, e.g., a service that can be used to instantiate a new MediaWiki. Therefore, in the third phase, the transformation from IPE models into initializable self-contained *Deployable Informal Process Essentials Archives (DIPEA)* takes place. We employ an *IPE Model Compiler*, which associates additional executables with resource definitions described by an IPE model. As a result, created DIPEAs enable enacting desired informal process.

Informal Process Model Deployment and Runtime (P4). In the final phase, the initialization of DIPEAs takes place. Therefore, we employ an *IPE Runtime* that is capable of parsing DIPEAs and running the executables contained in these archives. During this phase, the runtime acquires contained resources with desired interrelationships and initializes them. After acquiring resources, autonomous actors work towards intentions of informal processes collaboratively using other involved resources. For example, the runtime calls a service to initialize a MediaWiki and another service to assign an actor to a process via XING. Upon completion, the runtime releases all resources. For completeness, we include also the phase of modeling informal processes (P2). However, this phase is out of scope of this work and will be covered in future work.

4.2 Integrating Resources of Informal Processes (P1)

This section details the phase 1 by presenting a systematic approach to integrating resources into modeling and execution environments of informal processes. Different steps and components are involved in this phase as depicted in Fig. 2. To recap, this phase aims for establishing a modeling and execution environment of informal processes that is capable of creating and executing informal process models with *required* resources. As we aim in this phase for integrating resources involved in informal processes, we start with identifying resources that participate in informal processes (I1), as depicted in Fig. 2. During our scientific discussions, we have identified four resource domains that need to be investigated during this step: (i) *IT resources* including a wide range of applications and services to support actors in various ways such as a Redmine service and an automated build process of the respective software in the motivating scenario, (ii) *material resources* such as specific spare parts or tools to fix a machine, a simulation device, raw materials, etc., (iii) *knowledge resources* that contain the explicit knowledge of human actors, e.g., MediaWiki, Google Docs, a Model Repository, a Word document, etc., and (iv) *human resources*, human resources that are abstractly defined using roles, skills, and etc. The role of each domain becomes more critical depending on the nature of the informal process considered. For instance, in manufacturing, material resources gain more importance than the other resource domains, as it's more common to observe processes that are conducted manually with the help of material resources.

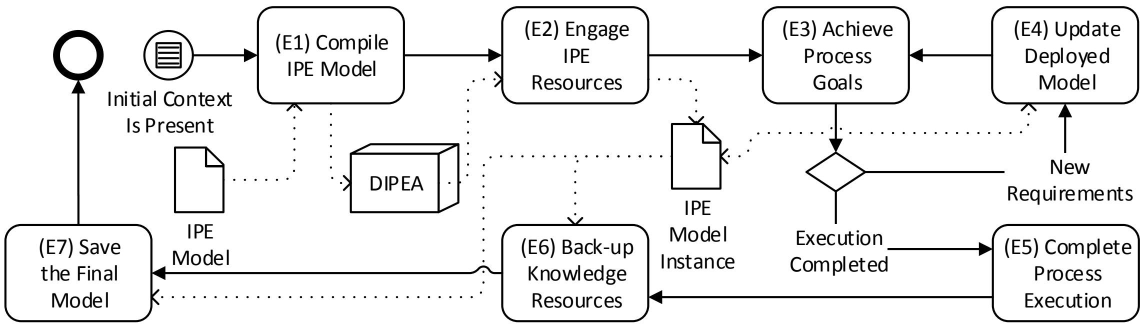
After identifying resources of interest, technical experts assess ways of integrating these resources into respective modeling and execution environments of informal processes. Therefore, they analyze available IT services that are related to the resources of interest (I2), e.g., they investigate relevant services for human resources such as XING. To this end, we distinguish between retrieval and control services. *Retrieval services* are capable of delivering information about different resources. Thus, using these services, it is possible to view available resources and their properties, which are used by business experts during modeling. For example, a social networking service enables viewing its members and their properties, i.e., human actors, of a social network. Another example is a business process repository that provides various IT resources such as the build plan of the software application from the motivating scenario. Moreover, each retrieval service can provide runnables, i.e., executables or *resource engagers*, to instantiate resources of informal processes. Examples of these runnables are scripts or business process models such as BPEL processes. These runnables typically communicate with different resource management services such as an operating system and a cloud service provider to acquire desired resources. Here, we opted for the word "engager", instead of, for example, "acquirer", as certain resources require an additional step of engaging towards informal process intentions. For instance, a resource engager of a human role can be in the form of a BPEL process that communicates with a human resource using a social network API during the acquirement, i.e., the engagement. Such a resource engager informs human actors and includes process intentions in the message so that actors can decide on joining the process and engage towards the intentions. As means of accessing different retrieval services can vary significantly, a direct integration of each retrieval service results in changes to an informal process modeling and execution environment. Moreover, such changes are not feasible in many cases and, as a result, a more loose-coupled way of the integration is required. So, technical experts implement *domain managers* to hide this heterogeneity and make use of retrieval services. Each domain manager abstracts from peculiarities of different retrieval services and provides a unified interface with equivalent operational behaviors. For instance, a domain manager of the social network mentioned would abstract the respective social network API by implementing certain operations with certain semantics such as listing available members of an organization. Moreover, it would provide the BPEL process mentioned for engaging the human resources. Thus, such a domain manager will act as a mediator and avoid any changes on an existing informal process modeling and execution environment. In certain cases, resources of a domain manager can be *inaccessible*, meaning that it is not possible or feasible to generate resource engagers for these resources. In such cases, technical experts can still develop domain managers only to represent the existence of resources. However, it is not possible to engage these resources in an automated fashion. Resource engagers are runnables and require runtime environments to be executed. The services that are capable of executing resource engagers are called *control services*. For example, a control service that is capable of running resource engagers in the form of BPEL processes is a BPEL engine. Different control services provide different APIs. Thus, integrating each control service into informal process execution environments requires changes in the respective execution environments themselves. To avoid such changes for different control services, technical experts hide control services behind *execution environment integrators*. Execution environment integrators are wrappers that provide access to different control services in a unified fashion. These integrators engage resources by running their executables during P4. We present a standard-based integration approach to realize these domain managers and execution environment integrators in our case study (Sect. 5) to validate the practical feasibility of this concept.

Although resources share common operations such as engaging resources for an informal process, there are also other important and relevant operations provided such as adding users to a MediaWiki and Redmine service. Before realizing domain managers and execution environment integrators, technical experts need to elaborate on required resource operations (I3). Thus, technical experts first analyze operations provided by different resources and their respective resource engagers. To this end, we distinguish between two kinds of operations (i) *life cycle operations*, e.g., acquire, release, get status, etc., and (ii) *custom operations*, e.g., send a message to a human resource. Life cycle operations enable initializing and releasing resources during the phase 4. Each resource engager must provide these operations to instantiate resources they represent. On the other hand, custom operations can be used for different objectives and vary for each resource selected. For example, "creating new user" operation of Redmine, i.e., giving a user certain access rights for using the corresponding resources, can be observed in many IT resources. In case all resources provided by a domain manager expose a certain operation, technical experts design domain managers and their respective execution integrators to support this operation. Moreover, there are also operations that belong to only a subset of resources provided by domain manager. For these operations, technical experts update respective resource engagers so that they provide necessary information about these extra operations.

After designing resource operations, they start realizing designed domain managers (I4). Hereafter, technical experts realize execution environment integrators to make the IPE Runtime support running different types of resource engagers (I5). After realizing each domain manager and execution environment integrator, technical experts register them to desired modeling and execution environments of informal processes (I6). Consequently, an aggregator component called *Resource Organizer* can collect available resources of different domains and present them to business experts for modeling. The registration data provides necessary information to access domain managers and execution environment integrators. After enacting different steps introduced in Fig. 2, business experts can use integrated resources during modeling, e.g., organizational roles provided by the domain manager of the XING service. Moreover, during execution, the IPE Runtime exploits the execution environment integrators created to instantiate IPE models, e.g., the execution environment integrator enacting a business process for engaging human actors via XING.

4.3 Compiling and Initializing Informal Process Models (P3 and P4)

As a preliminary phase of enacting informal processes, business experts model existing informal processes in their organizations (P2). Modeling of informal processes is followed by executing informal processes. Although we consider the compilation phase (P3) and the deployment and runtime (P4) separately in our overview, they both constitute to the automated execution of informal processes and, thus, are considered to be logically grouped here. Typically, initializing informal process models starts after the initial context defined in an IPE model is present in the respective organization, e.g., the need for developing a new service in the motivating scenario. Either an automated service, e.g., an activity in a structured process, recognizes the presence of an initial context definition and triggers the enactment automatically or the respective organization triggers it manually. The triggering is followed by compiling an IPE model. IPE



(E1) Compile IPE Model
(E2) Engage IPE Resources
(E3) Achieve Process Goals
(E4) Update Deployed Model
Initial Context Is Present
DIPEA
IPE Model
IPE Model Instance
New Requirements
Execution Completed
(E7) Save the Final Model
(E6) Back-up Knowledge Resources
(E5) Complete Process Execution

Figure 3: Executing Informal Processes

models do not comprise required runnables for the resources to be acquired, but, rather, they provide only descriptive information about them. The compilation step (E1) converts the IPE model with its descriptive nature into a *Deployable Informal Process Essentials Archive (DIPEA)*, which contains all required runnables, e.g., scripts, business processes, etc, for initializing an informal process model. During this step, an IPE Model Compiler interacts with domain managers of *accessible* IPE model resources to collect resource engagers. For instance, compiling an IPE model containing a human actor results in an archive that contains a BPEL process for engaging that actor via XING. After the compilation, the execution proceeds with acquiring resources of the IPE model (E2). During the acquirement, the IPE Runtime parses the DIPEA and runs each resource engager contained using its associated execution environment integrator. For instance, a DIPEA containing a human role and its respective resource engager in the form of a BPEL process is executed using its execution environment integrator of a BPEL engine, respectively. Failing the initialization of any of the *initial* resources leads to a faulty state.

The successful initialization results in an *IPE Model Instance*, as depicted in Fig. 3. A model instance contains additional meta-data about executed processes such as the information about the start time, a history of the resource model, the time of changes made, etc. Moreover, each resource definition is converted into *resource instances*. Resource instances contain the necessary information to access or track a resource or its activities, respectively. For instance, a resource instance of a MediaWiki resource residing in a cloud infrastructure contains its respective IP address and required credentials to execute desired operations, e.g., add a new user. Similarly, a resource instance representing a human actor described by a role resolves into an actual individual identified by its ID in the XING service. After completing the initialization, engaged actors work towards informal process intentions using other acquired resources (E3). Human actors can update the initialized informal process model to adapt the process execution to emerging requirements (E4). Adaptations can mean changing existing resource models, intentions, and the associated context information. For instance, business experts can acquire the service to manage IoT devices in the motivating scenario during E3 automatically by executing its resource engager, e.g., a BPEL process.

After achieving informal process goals, completing the process execution starts (E5), e.g., the completion of the software development process. During this step, the IPE Runtime releases each resource using their execution environment integrator. For example, the resources that have been provisioned in a cloud environment are de-provisioned. During releasing knowledge resources, special care must be taken. As these resources must evolve iteratively, they must be stored and reused after each execution (E6). Each back-up operation is a resource-specific operation meaning that they are defined separately in respective resource engagers. For example, storing the knowledge contained in a MediaWiki is different than the knowledge contained in a simple Word document. After finalizing backups, the IPE Runtime stores the final state of an informal process instance (E7). These different versions provide an overview of different possible intentions and resource structures resulted during different executions of the same process. Business experts can select one of the versions for future executions based on different parameters such as the execution time, the cost, or the frequency of the model execution.

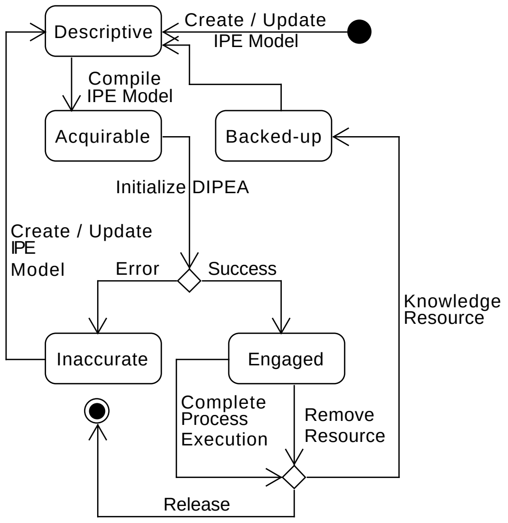


Figure 4: Life Cycle of a Resource During Informal Process Execution

During the execution, each resource goes through a certain life cycle, as depicted in Fig. 4. Each resource appears firstly in a *descriptive* form in IPE models. From these descriptive resource models, IPE Model Compilers create *acquirable* resources using their respective domain managers. These are still abstract resources that need to be resolved during the acquirement. For example, resource engagers do not contain the information about an actual employee, but, rather, just roles. Similarly, software licenses or other computational resources are not yet acquired for acquirable IT resources. After the initialization step, the execution environment resolves actual resources. If the acquirement of a resource has failed, the state changes into *inaccurate*, meaning that the created deployment descriptor of the resource is not accurate, as denoted in Fig. 4. For example, the resolution of human roles can fail during deployment, as no human was present with the desired role. After models are acquired, they are in the state *engaged*, as they work towards the goals of an informal process. Changing requirements can result in updates in resources of informal processes, which will follow the same state diagram explained previously. Accomplishing goals results in completing the respective process and releasing the acquired resources. In case the resource to be released is a knowledge resource, it's backed up for the next execution.

5. CASE STUDY USING OPENTOSCA AND APIFICATION

To asses the feasibility of the approach presented, we develop a case study centered around our motivating scenario using the DevOps knowledge base [25], the OpenTOSCA ecosystem (Sect. 5.1), and the APIfication approach (Sect. 5.2). By including OpenTOSCA in our case study, we can create pre-packaged resources for clouds infrastructures, e.g., MediaWiki. Moreover, through the use of APIfication approach we enable reusing any existing DevOps artifact such as a script for installing Redmine on a server contained in the knowledge base. Various deliverables of the case study are provided as *Supporting Online Material (SOM)*[6]. In the following, we firstly describe fundamentals of our case study: the OpenTOSCA ecosystem and the APIfication approach. Thereafter, we present InProXec framework and its application.

5.1 OpenTOSCA

Many informal processes are executed using IT resources, e.g., software services such as knowledge management services, simulation services, software development solutions, etc. Such IT resources can be automatically provisioned on-demand using *cloud computing* technologies [23] and can be automatically provisioned on-demand via self-service portals using cloud computing technologies [15]. The spread of cloud provider offerings has urged the need of a standardization for the interoperability of cloud applications. This need has resulted in the creation of the Topology and Orchestration Specification for Cloud Applications (TOSCA) standard [3]. TOSCA describes cloud services in two parts: (i) An application topology model of a service contains different components of an application and their interrelationships. (ii) The management logic of services contains the information about how to execute management operations of the respective application topology. The management logic is described in the form of business processes. After preparing these two parts, the application topology and the management logic of cloud applications are stored in a self-contained archive called Cloud Service Archive (CSAR). CSARs are later deployed on TOSCA runtime environments such as OpenTOSCA [2]. TOSCA containers use the topology model and the available management processes to instantiate the described application and to manage them, e.g., to scale application components. The result of the deployment is the provisioned application in the desired cloud infrastructure. The OpenTOSCA ecosystem provides a set of tools for modeling, deploying, and executing services created using TOSCA. It comprises three major components, (i) Winery [13], (ii) OpenTOSCA Container [2], and (iii) Vinothek [4]. Winery is a web-based modeling tool to create TOSCA models that are deployed on a TOSCA-complaint container, e.g., the OpenTOSCA Container. Vinothek is a self-service portal that can be used by humans to trigger the instantiation of services. In this work, we reuse this toolchain in our case study to acquire IT resources in cloud infrastructures.

5.2 APIfication

A major building block and enabler for our approach is to generate APIs for arbitrary executables, i.e., the *APIfication* of executables such as deployment scripts, compiled programs, configuration definitions, etc. [26]. We employ this concept to wrap resource engangers of different implementations as uniform APIs that can be easily accessed by the human actors and we call this step as the APIfication of resouce engagers. Such a generated API implementation wraps an executable and exposes its functionality, e.g., as RESTful or SOAP-based Web service in a self-contained manner. These self-contained API implementations are packaged using a portable mechanism such as a Docker container or a Vagrant VM build plan. Consequently, generated API implementations can run in various environments such as a developer's laptop, a test cloud, or a production cluster. The type of the generated interface (REST, SOAP, JSON-RPC, etc.) is selected depending on the constraints of the environment such as existing expertise, developer's preferences, or the overarching orchestration technique. For instance, if BPEL workflows would be used for orchestration, a SOAP-based Web service interface would be the preferred choice. We utilize ANY2API [26] as an APIfication framework to generate API implementations for arbitrary executables. The framework is completely modular and extensible. Invoker modules (e.g., a *Python invoker*) provide and encapsulate the invocation logic for a certain kind of executable, e.g., Python scripts. Generator modules (e.g., a *REST API generator*) implement the functionality to generate and package an API implementation based on a given API specification (*API spec*[7]). Such an API spec defines the interface of an executable (invocation command, mapping of parameters and results to files, environment variables, `STDIN`, `STDOUT`, etc.) in a generic manner that is not bound to any particular kind of API. Consequently, the packaged API implementation can be treated as a black box, so users of the API do not have to know anything about the technical internals of the API implementation and the wrapped executable. Instead, a proper API is provided to the user to ease the interaction with the packaged executable. To improve efficiency and to reduce resource consumption at runtime, multiple executables of different kinds can be packaged together with the required invokers in a single API implementation [26]. In this paper, we use this approach to create resource engagers with unified APIs.

[6]http://www.co-act.biz/downloads/

[7]API spec documentation: http://any2api.org/apispec

5.3 InProXec Framework

We are implementing a framework to realize the InProXec method presented. The framework enables integrating resources of informal processes using a *client integration library* into modeling and execution environments of informal processes (P1). This client library eases the adoption of existing resource retrieval services and enables managing resources provided by different services in a unified fashion. Moreover, we introduce necessary components for compiling and initializing informal process models (P3 and P4). In the following, we present our core system to realize the phases 1, 3, and 4 of the InProXec method.

We start our case study with implementing the following components: (i) the Resource Organizer, (ii) the IPE Model Compiler, (iii) and the IPE Runtime as they are the enablers of modeling and execution. We develop each component as a service and we expose most of their functionalities aligned with WS-* standards[8]. As a result, components are modular, standard complaint, and composable. The resource organizer service is responsible for collecting resource definitions from the domain managers integrated in the phase 1 and providing them, e.g., for modeling. The organizer polls the definitions from subscribed domain managers every 30 seconds, aggregates collected ones in a data store, and provides them upon request. As both domain managers and the resources provided are identified uniquely, the Resource Organizer service can store each resource definition without any collisions. As an exchange format among different components, we opted for the Web Ontology Language (OWL) to enable semantic discoveries on informal process entities. The most recent version of this ontology can be obtained in the provided SOM. To ease developing domain managers of different retrieval services, we use a client integration library. The client library subscribes domain managers using the metadata each domain manager provides. Domain managers of *accessible* resources, i.e., resources that can be acquired automatically, provide also a resource engager, e.g., a script, business process, etc., for the resource initialization. The client integration library enables collecting resource engagers of accessible resources over a REST API. It automatically enriches resource definitions of each domain manager with URIs specifying means of accessing corresponding resource engagers As a result, the IPE Model Compiler can collect them during the compilation. For instance, the client library enriches the resource definition of MediaWiki with additional deployment information about the location of its resource engager, i.e., its CSAR. Thereafter, the IPE Model Compiler, which is responsible for creating DIPEAs, collects all resource engagers of the resources contained using the REST API of the integration client. For example, during the compilation of an IPE model containing a MediaWiki resource, the IPE Model Compiler service fetches the MediaWiki CSAR using the URI found in its resource definition.

The IPE Runtime service acquires resources contained in a DIPEA using execution environment integrators. Similarly, technical experts can use the integration library that we develop to integrate execution environment integrators. After the registration, the access details of execution environment integrators are stored in a central repository with their metadata. The IPE Runtime interacts with execution environment integrators using the REST API of the integration client provided to run resources engagers packed in DIPEAs. The runtime service initializes a business process, e.g., a BPEL process, for instantiating IPE models. Thus, we reuse various concepts that were developed for structured processes such as managing parallel process instances.

[8]http://www.w3.org/2002/ws/

5.4 Integrating Resources Of Informal Processes (P1)

In this section, we apply the steps of integrating resources of informal processes introduced in the first phase of our method to the motivating scenario. The first step in this phase aims for identifying resources involved in informal processes (I1). In the motivating scenario, it's easy to spot the existence of different resources from IT and human resource domains. The following focuses on the domain of IT resources, as the case study is built on top of the OpenTOSCA system, the DevOps knowledge base, and the APIfication approach. Moreover, integrating human resources requires custom-tailored resource engagers, which will be covered in future work. After identifying required resources of informal processes, e.g., MediaWiki and Redmine, technical experts analyze retrieval and control services of these resources (I3). As Winery provides a MediaWiki service, it is one of the retrieval services. Moreover, Winery can provide also other pre-packaged cloud services of an organization. Additionally, the rest of the DevOps artifacts such as a script for creating a Redmine service is stored in the DevOps knowledge base. The OpenTOSCA container provides a runtime environment for resource engagers generated by Winery, i.e, CSARs. As the knowledge base delivers resource engagers with heterogeneous APIs, we create standardized Docker compatible resource engagers through the APIfication approach, i.e., the APIfication of resource engagers. Thus, the identified control services are (i) the OpenTOSCA container for CSARs and (ii) Docker for the knowledge base artifacts. Identifying control and retrieval services is followed by a designing resource operations (I3). During this step, apart from life cycle operations such as acquiring resources, we have determined that MediaWiki and Redmine have user management operations in common. However, as these resources are provided by different domain managers, i.e., domain managers of Winery and the knowledge base, respectively, we decided to model these operations at the level of resource engagers. Thereafter, we develop execution environment integrators of the OpenTOSCA container and Docker using the client integration library to enable acquiring resource engagers in the form of CSARs and the Docker compatible artifacts created through the APIfication approach (I5). Developing the managers and the integrators is followed by registering each execution environment integrator and domain manager by calling register operation of the provided client integration library (I6).

5.5 Compiling and Initializing Informal Process Models (P3 and P4)

This section presents the application of the phases 3 and 4 of the method (Sect. 4.3) to the motivating scenario. After configuring informal process modeling and execution environments, business experts can create IPE models using the integrated resources in P1, e.g., MediaWiki and Redmine. Moreover, the IPE Runtime service can initialize these models. To realize different steps of phase 3 and 4, we develop an *execution process*, e.g., a BPEL process, that enacts the steps illustrated in Fig. 3. The IPE Runtime service initializes an IPE model by initiating this execution process. Firstly, the process uses the IPE Model Compiler service for generating corresponding DIPEA from the IPE model received (E1). For instance, an IPE model containing Redmine and MediaWiki is converted into a DIPEA with a MediaWiki CSAR and a Docker compatible Redmine script. Hereafter, the execution process acquires resources defined in the IPE model by parsing the generated DIPEA with the help of the IPE Runtime service (E2). The IPE Runtime service uses corresponding execution environment integrators for running resource engagers. For example, a resource engager in the form of a CSAR for MediaWiki is run by the execution environment integrator of the OpenTOSCA container. Acquired resources work towards intentions of the respective informal process (E3). Hereafter, business experts update the list of resources using an instance management window, e.g., an IoT middleware (E4). After achieving intentions of an informal process (E5), the process execution proceeds with the release of provisioned resources. Knowledge resources, e.g., MediaWiki, provide an additional operation that results in backing up of respective knowledge resource. The process is completed by calling the IPE Runtime service. The IPE Runtime stores knowledge resources in the form of a resource engager and reuses it during the next execution. Thus, each back up operation returns an updated resource engager for that specific resource, e.g., backup operation of the MediaWiki returns an updated MediaWiki CSAR containing the current data of the MediaWiki. Finally, the final state of IPE instance model is saved (I7).

By applying the InProXec method on our case study, we prove the applicability of our method. The method enables acquiring and engaging resources and actors in informal processes in an automated fashion. As a result, the method eliminates the manual work required for establishing the environment to complete informal processes, e.g., creating a MediaWiki service, assigning different actors, etc. Moreover, the concept of knowledge resources preserve the explicit knowledge created and support actors during future enactments, e.g., the knowledge contained in the MediaWiki service. Reusing existing and working components strengthens our case study and ease the adoption of the approach by different organizations. To ease creating domain managers and execution environment integrators, we include a client integration library for in the provided SOM. As a consequence, we enable an easier adoption of our approach. We are continuously refining the modeled resources and processes to make this case study an on-going one, considering various additional aspects such as relationships and additional resources. By providing a WSDL API for the framework, we provide an easy integration to existing structured process modeling languages such as BPEL. Moreover, activity-oriented approaches such as adaptive case management [12] or ad-hoc process management [10] can trigger an IPE model upon initialization.

6. RELATED WORK

Different approaches have been proposed to model and automate business processes. Activity-centric approaches [10, 16, 12, 5, 14] enable designing processes based on their structured activity sequences. There are approaches to (i) modeling processes beforehand and adapting them on the fly [5, 14] (ii) and creating process activities on the fly to capture the activity structure for reusing during next executions [10, 16, 12]. Using BPEL4People[9] and WS-HumanTasks[10], human activities can be integrated into automated business processes. Moreover, Barukh and Benatallah [1] introduce a hybrid-process management platform for integrating structured, semi-structured, and unstructured activities in a process management platform. As informal processes are unstructured processes, the reusablility of activity-centric process models decreases. Actors of informal processes enact them based on their knowledge and experience complaint with certain constrains defined by respective organization [6]. The Subject-Oriented Business Process Management (S-BPM) approach [11, 18] aims for modeling business processes based on “subject”, “predicate”, and “object” triplets. Models created using this approach are at a higher level of abstraction and require further refinements for executing them in an automated fashion. They represent interactions among different actors and responsibilities of these actors. Van der Aalst et al. [22] follow a declarative approach and focus on constrains of processes. Actors can execute the desired activities as long as they stay in the previously defined constrains of the respective process. Moreover, goals of each informal process are known initially [6]. Thus, a goal-oriented approach can be used to model informal processes. Nurcan [17] proposes such a goal-oriented approach to define higher level processes, i.e., processes that define higher-level activities in an organization.

Informal processes are collaborative meaning that participants of an informal process collaborate with each other to accomplish its respective goals. Designing these collaborations play a critical role during enacting the respective informal processes. To design collaborations, one can use collaboration patterns such shared object or publish / subscribe [7]. Using these patterns, one can define the architecture of a collaboration with modeling approaches such as human Architecture Description Language [8]. However, these collaborations are typically at a higher level of abstraction and needs to be refined before deploying them. Therefore, in our previous work [19], we present a transformation from collaboration architectures into informal processes. Using this transformation an abstract architectural model describing a collaboration can be converted to an informal process, which can be deployed by the method detailed in this paper.

7. CONCLUSION AND OUTLOOK

Automating informal processes require engaging qualified actors towards goals of the process in an automated fashion. Automatically acquired actors can autonomously accomplish these goals with the help of other involved resources. In this work, we present a method for automating the initialization of resource-centric informal processes. To this end, we present two steps: (i) integrating resources of informal processes and (ii) executing informal processes. The integration method empowers integrating resources of informal processes into informal process modeling and execution environments. Thereafter, different components such as a model compiler and a runtime component initialize informal process models with integrated resources of informal processes. The approach is validated using a case study, which uses

[9]http://docs.oasis-open.org/bpel4people/bpel4people-1.1.pdf

[10]http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cs-01.pdf

existing components such as the knowledge repository, the OpenTOSCA ecosystem, and the APIfication approach.

This work limits its scope to resource models without relationships due to its existing broad scope. Therefore, in future work, we will investigate deploying interrelated resource models. Moreover, we are investigating a modeling approach for informal processes, which is supported by a graphical notation.

8. ACKNOWLEDGMENTS

This work has been partially supported by Graduate School of Excellence advanced Manufacturing Engineering (GSaME)[11] and SitOPT (Research Grant 610872, DFG).

9. REFERENCES

[1] M. Barukh and B. Benatallah. ProcessBase: A Hybrid Process Management Platform. In *Service-Oriented Computing*. Springer Berlin Heidelberg, 2014.

[2] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner. OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In *ICSOC'13*, 2013.

[3] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann. TOSCA: Portable Automated Deployment and Management of Cloud Applications. In *Advanced Web Services*. Springer New York, 2014.

[4] U. Breitenbücher, T. Binz, O. Kopp, and F. Leymann. Vinothek - A Self-Service Portal for TOSCA. In *Proceedings of the 6th Central-European Workshop on Services and their Composition, ZEUS 2014*, 2014.

[5] P. Dadam and M. Reichert. The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support - Challenges and Achievements. *CSRD*, 23:81–97, 2009.

[6] C. Di Ciccio, A. Marrella, and A. Russo. Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary Approaches. *JoDS*, 2014.

[7] C. Dorn and R. Taylor. Analyzing runtime adaptability of collaboration patterns. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, 2012.

[8] C. Dorn and R. Taylor. Architecture-driven modeling of adaptive collaboration structures in large-scale social web applications. In *Web Information Systems Engineering – WISE 2012*. Springer Berlin Heidelberg, 2012.

[9] M. Dumas, W. M. Van der Aalst, and A. H. Ter Hofstede. *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, 2005.

[10] S. Dustdar. Caramba–A Process-Aware Collaboration System Supporting Ad-hoc and Collaborative Processes in Virtual Teams. *Distributed and Parallel Databases*, 2004.

[11] A. Fleischmann. What is s-bpm? In H. Buchwald, A. Fleischmann, D. Seese, and C. Stary, editors, *S-BPM ONE – Setting the Stage for Subject-Oriented Business Process Management*, volume 85 of *CCIS*, pages 85–106. Springer Berlin Heidelberg, 2010.

[12] C. Herrmann and M. Kurz. Adaptive Case Management: Supporting Knowledge Intensive Processes with IT Systems. In *S-BPM ONE 2011*. Springer Berlin Heidelberg, 2011.

[13] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann. Winery – Modeling Tool for TOSCA-based Cloud Applications. In *ICSOC'13*, 2013.

[14] A. Marrella, A. Russo, and M. Mecella. Planlets: Automatically recovering dynamic processes in yawl. In *On the Move to Meaningful Internet Systems: OTM 2012*. Springer Berlin Heidelberg, 2012.

[15] P. Mell and T. Grance. The NIST definition of cloud computing (draft). *NIST special publication*, 800:7, 2011.

[16] P. Moody, D. Gruen, M. Muller, J. Tang, and T. Moran. Business activity patterns: A new model for collaborative business applications. *IBM Systems Journal*, 45:683–694, 2006.

[17] S. Nurcan, A. Etien, R. Kaabi, I. Zoukar, and C. Rolland. A strategy driven business process modelling approach. *BPMJ*, 11:628–649, 2005.

[18] R. Singer and E. Zinser. Business Process Management – S-BPM a New Paradigm for Competitive Advantage? In H. Buchwald, A. Fleischmann, D. Seese, and C. Stary, editors, *S-BPM ONE – Setting the Stage for Subject-Oriented Business Process Management*, volume 85 of *CCIS*, pages 48–70. Springer Berlin Heidelberg, 2010.

[19] C. Sungur, C. Dorn, S. Dustdar, and F. Leymann. Transforming Collaboration Structures into Deployable Informal Processes. In *Engineering the Web in the Big Data Era*. Springer International Publishing, 2015.

[20] C. T. Sungur, T. Binz, U. Breitenbücher, and F. Leymann. Informal Process Essentials. In *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*. IEEE Computer Society, 2014.

[21] C. T. Sungur, O. Kopp, and F. Leymann. Supporting Informal Processes. In *Proceedings of the 6th Central-European Workshop on Services and their Composition, ZEUS 2014*, 2014.

[22] W. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *CSDR*, 23:99–113, 2009.

[23] K. Vukojevic-Haupt, D. Karastoyanova, and F. Leymann. On-demand provisioning of infrastructure, middleware and services for simulation workflows. In *SOCA 2013*, pages 91–98. IEEE Press, 2013.

[24] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer New York, Inc., Secaucus, NJ, USA, 2007.

[25] J. Wettinger, V. Andrikopoulos, and F. Leymann. Automated Capturing and Systematic Usage of DevOps Knowledge for Cloud Applications. In *Proceedings of the International Conference on Cloud Engineering (IC2E)*. IEEE Computer Society, 2015.

[26] J. Wettinger, U. Breitenbücher, and F. Leymann. Any2API - Automated APIfication. In *Proceedings of the 5th International Conference on Cloud Computing and Services Science*. SciTePress, 2015.

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