Transforming Collaboration Structures into Deployable Informal Processes

C. Timurhan Sungur[1(✉)], Christoph Dorn[2], Schahram Dustdar[2], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany
{sungur,leymann}@iaas.uni-stuttgart.de
[2] Distributed Systems Group, Vienna University of Technology, Vienna, Austria
{dorn,dustdar}@dsg.tuwien.ac.at

Abstract. Traditional workflow and activity-centric coordination offers limited process support to human collaborators when unanticipated situations predominate. Under such circumstances, informal processes focus on provisioning relevant resources for achieving collaboration goals. Resources include interaction mechanisms such as shared artifact, social networks, and publish/subscribe information dissemination as complex situations typically demand the close collaboration among multiple human process participants. Currently, however, there exists a gap between (i) selecting and configuring suitable interaction mechanisms (collaboration level) and (ii) deploying the respective collaboration platforms (IT level). In this paper, we present an approach and techniques for transforming collaboration structures into automatically deployable informal processes. We demonstrate how our tools support the specification of desirable collaboration capabilities subsequently deployed to multiple MediaWiki instances.

Keywords: Informal process essentials · Human architecture description language · Wiki · Collaboration configuration · Transformation

1 Introduction

Organizational processes range from ad-hoc activities to rigorously-defined workflows [19]. As flexibility and adaptation is required for unforseen situations, humans tend to execute most activities in those less well-defined processes—denoted in this paper as *informal processes*. With fewer a-priori specified flow conditions, however, comes the need for supporting coordination and collaboration among human process participants. Informal processes are hence inherently human-centric. Here, the traditional process modeling primitives (i.e., elements for tasks, control flow, data flow, etc) are often no longer suitable [15,30]. Other interaction dependency management patterns [17] such as *Shared Artifact*, *Social Network*, *Secretary/Principal*, *Master/Worker*, or *Publish/Subscribe* become increasingly relevant. Collaboration patterns allow modeling how multiple humans interact through messages, artifacts,

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 231–250, 2015.
DOI: 10.1007/978-3-319-19890-3_16

requests, etc, rather than having to express all work efforts by means of assigning a single task to a single process participant.

Multiple platforms implement the various collaboration patterns. Facebook and LinkedIn, for example, realize the Social Network pattern; MediaWiki [1], DokuWiki [2], and Google Documents realize the Shared Artifact pattern, Twitter the Publish/Subscribe pattern. Informal processes target unforeseen situations and thus require flexible coordination mechanisms. It is highly unlikely that, for example, a given wiki or social network platform in its initial, static configuration will remain suitable for all informal process purposes. We therefore argue that in human-intensive process environments, we need to treat coordination and collaboration mechanisms as resources that are specifically configured for each process instance. Statically provided collaboration mechanisms cause two types of problems: on the one hand, they may prove too flexible (e.g., pure email) and thus cannot properly support the process participants. Participants need then to execute all coordination actions manually without any monitoring, notification, protection, or other automation support. On the other hand, static configurations may lead to overly rigid coordination structures that quickly become inefficient to use or are abandoned altogether. An example are traditional workflow systems that exactly define who must do what in which order.

Providing collaboration mechanism as a configurable resource, however, entails a set of challenges. Typical resource management and provisioning mechanisms focus on fair, safe, or cheap access by individual clients [27]. Access privileges specify what a single client may or may not do with a particular resource. Collaboration mechanism resources, in contrast, are intrinsically subject to simultaneous, correlated use. Furthermore, collaborators enact process-centric roles that need finely-tuned, highly asymmetric usage capabilities (e.g., reading vs. writing, sending vs. receiving). While process participants are aware of these fine-grained collaboration-level requirements, IT personnel managing the actual resource have only insights into how to configure and deploy the corresponding software.

Our contribution in this paper are a set of models, transformations, and supporting framework for addressing these challenges. Our approach separates the specification of collaboration roles [16] from their provisioning details [30]. Along these lines, informal process participants utilize a collaboration-centric architecture description language for determining the required roles and collaboration actions for their particular informal process instance. This specification is tied to implementation artifacts via resource capabilities. Automatic deployment of implementation artifacts closes the gap between collaboration and IT level. We demonstrate the feasibility of specifying and deploying non-trivial collaboration structures based on a use case involving multiple wikis.

The remainder of this paper is structured as follows. Section 1.1 introduces an accompanying scenario. We subsequently provide background on applied models and techniques (Sec. 2) followed by an overview of our approach (Sec. 3).

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

Section 4 describes the transformation and refinement process for mapping the collaboration specification to the providing software artifacts and their deployment. We demonstrate and discuss this procedure within the scope of a validating use case (Sec. 5). Section 6 compares related work to our approach before Section 7 completes this paper with an outlook on future work and conclusions.

1.1 Motivating Scenario

Suppose a research consortium envisions the knowledge collection and dissemination from project partners as well as the wider research community. Project members are, however, interesting in keeping preliminary articles on work in progress separated from stable, generally publishable results. The differences in the expected user base and work coordination call for significantly different user roles and access permission for each article type. The various ways people collaborate and engage in article authoring, quality management, commenting, and improving renders futile any attempt to adequately capture these in a traditional workflow-style model. Such a process model will exhibit rigid action sequences and inhibit ad-hoc collaboration. In most such environments, custom-made software is neither an option due to time and/or cost constraints. Resorting to pre-existing solutions such as a wiki is a step in the right direction. A standard wiki provides a suitable set of collaboration capabilities but rarely exhibits the right capability configuration by default. Given the permission model of a typical wiki software (e.g., MediaWiki), the scenario's article differences demand two wiki instances, each configured accordingly (see Fig. 1 left).

One potential configuration for an internal wiki instance may foresee user authentication even for merely reading articles. Editing and managing roles are kept simple and to a minimum. Each project participating organization allocates one quality manager who is also responsible for promoting regular users to editors. Editors obtain generous editing rights. For example, new articles are automatically approved; quality managers merely check changes to existing articles for sensitive data from time to time.

The external wiki, on the other hand, needs more sophisticated users groups. Specifically, the configuration needs to balance low entry barriers for consuming and contributing knowledge with measures targeting article spamming, vandalism, and edit wars. To this end, the wiki might allow anonymous read access, a basic set of editing rights for authenticated users, extensive editing rights for time-trusted experts, and separate roles for user management, article patrolling, and article protecting.

The project's members are only interested in determining the wiki configuration and understanding its implications at the collaboration level (in contrast to IT level configuration concerns such as security, persistence, and scalability). The following sections describe how to operationalize such separation of concerns.

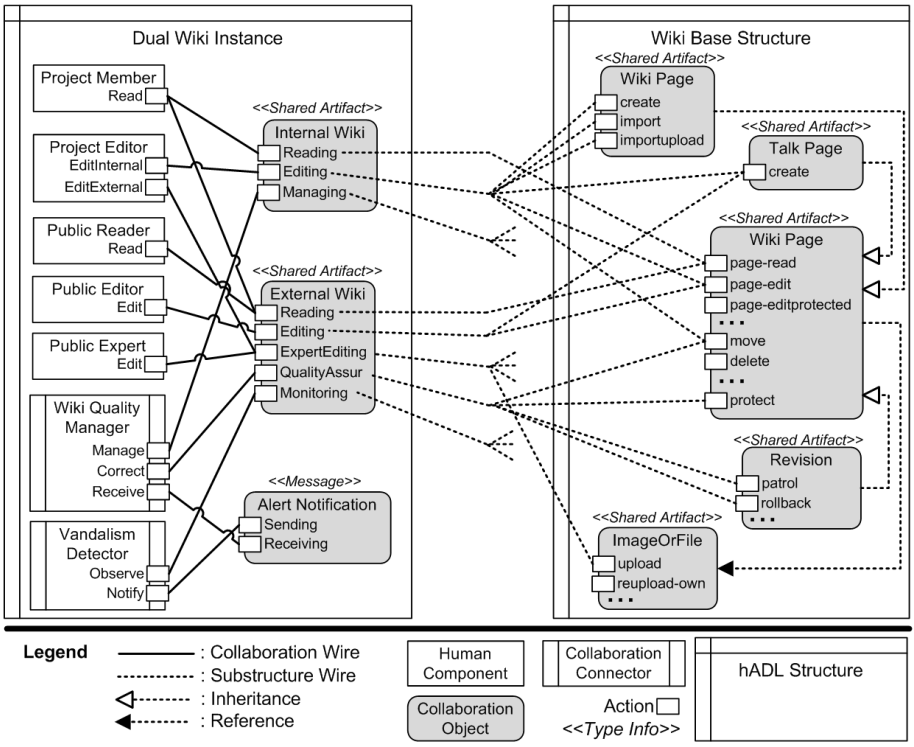

Fig. 1. Dual wiki scenario with substructure mapping in schematic hADL

2 Background

2.1 Human Architecture Description Language

The human Architecture Description Language (hADL) [16] provides a collaboration-centric component and connector view. A hADL model describes the collaboration structure in terms of the interacting user roles and their available interaction mechanisms. To this end, hADL distinguishes between *HumanComponents* and *CollaborationConnectors* to emphasize the difference between the primary collaborating users and non-essential, replaceable users that coordinate the collaboration. In our motivating scenario (Fig. 1 far left), project members, project editors, public read, public editor and public expert roles are the HumanComponents. The Wiki Quality Manager and Vandalism Detector roles represent CollaborationConnectors. They aren't strictly necessary, but greatly facilitate collaboration. A CollaborationConnector is thus responsible for the efficient and effective interaction among HumanComponents. It thereby may cover the full automation spectrum: from purely human, to software-assisted, to purely software implemented.

Users employ diverse means of interaction that range from emails, to chat rooms, shared wiki pages, and Q&A forums, to vote collection. These means implement vastly different interaction semantics: a message is sent and received, a shared artifact is edited, a vote can be cast. *CollaborationObjects* abstract from concrete interaction tools and capture the semantic differences in subtypes, e.g., *Message*, *Stream*, or *SharedArtifact*. In our example, the two *Wikis* (Fig. 1 mid left) provide collaboration in the form of a shared artifact, while the *Alert notification* provides messaging-centric capabilities. The actual notification mechanism may then be implemented through email, XMPP, SMS, or even a combination thereof.

Actions specify what capabilities a component or connector requires to fulfill his/her role, e.g., edit an article or receive an alert message. Complementary, *Actions* on CollaborationObject determine the offered capabilities. To this end, actions distinguish between *Create*, *Read*, *Update*, and *Delete* (CRUD) privileges. Action cardinalities further specify the upper and lower boundaries on the number of collaborators which may simultaneously have acquired the action's capabilities. The *Alert Notification Receiving* action, for example, thus demands at least one component or connector having receiving privileges when exhibiting an action cardinality of *(1..*)*.

Ultimately, *Collaboration Links* connect HumanComponent and CollaborationConnector actions to CollaborationObject actions, thus wiring up a particular collaboration structure. The *Structure* element provides a containment mechanism for complex, hierarchical CollaborationObjects and interaction patterns composed from the basic hADL elements. The scenario depicts the use of substructures for detailing the internal structure of each wiki collaboration object. The parent element references a pre-existing structure (here the Wiki Base Structure), and provides a mapping between parent action (e.g., Internal Wiki - Editing) and substructure action(s) (here Wiki Page - create, import ...) via substructure wires (Fig. 1 center, dashed lines). Multiple substructure wires between a single parent action and multiple sub actions imply *aggregation* semantics. Hence, the Internal Wiki editing action aggregates the Wiki Page create, import, import-upload, Page page-edit, etc capabilities. Likewise, two substructure wires from different parent actions to the same sub-action imply capability reuse, i.e., both parent actions make the sub-action's capability available. The substructure mechanism thus allows different configuration of the same base structures. Internal Wiki and External Wiki only need to exhibit the desired substructure wiring without having to duplicate the Wiki Base Structure.

2.2 Informal Process Essentials

Business process modeling languages, e.g., BPEL [26] and BPMN [2], enable capturing recurring activity patterns in various domains, e.g., manufacturing, health-care, IT, etc. Consequently, the activities, which frequently occur, can be documented, re-executed, and further improved. Typically, these activity-oriented process modeling approaches focus on the repeated activities and their structure. On the other hand, there are human-centric informal processes

which cannot be well-defined using these activity-oriented approaches as their activities and sequences constantly change. Rather than using activity-oriented approaches, one captures the essential repeated information using resource-centric approaches such as Informal Process Essentials (IPE) [30]. The desired process result may be repeatedly obtained through selection of the same set of resources (and subsequently engaging them towards the collective goals of informal processes). IPE's resource-centric approach follows an agent-oriented style as an IPE model may specify resources that represent any type of active entities (humans, services, hardware, etc.) which then work autonomously towards the desired intentions of the respective informal process. Each IPE model contains the list of necessary resources to accomplish the target goal of the corresponding informal process. Each resource may exhibit various semantical relationships with other resources for specifying complex resource sets. These resources are typically provided by some services, i.e., resource organizers. Resource organizers are responsible for resource life-cycle operations of the resources within the corresponding process scope, i.e., they prepare the resources for the process execution and they release them upon process completion. Informal process actors are a special resource type. They work autonomously and make use of all other provided resources within the scope of their informal process instance. Resource organizers are added to the respective system on a plug-in basis. Each plug-in is responsible for the respective resource domain, e.g., a resource organizer plug-in for OpenTOSCA ecosystem. The combination of a domain specific resource organizer and its adapter results in a pluggable resource organizer. In the scope of this work, whenever we mention a specific resource organizer, we implicitly include its adapter.

An IPE model describes the the main intention that reflects the informal process' main goal. Each intention may be refined through sub-intentions that also may serve as constraints, e.g., "complete the process in one day". IPE model enactment depends on conditions in the surrounding execution environment. The IPE model's initial context specifies the triggers that signal when the model's corresponding resources should be initialized and subsequently work towards the informal process' main goal. Complementary to the initial context, the IPE model specifies the resulting context which specifies the conditions for determining the processes' main intention as successfully achieved. The expression of initial and final context is out of the scope of this work and will not be further discussed.

Fig. 2 illustrates the IPE Model for the motivating scenario. The IPE's collaborative resource specification is derived from the hADL architecture of the scenario. It details the various human performers who use the Wiki software in terms of their particular roles and permissions. The resources, that ultimately enact the collaboration roles, are organized by resource organizers, e.g., IT resources are organized by OpenTOSCA plug-in communicating with the OpenTOSCA ecosystem which is a cloud application management and deployment container. The main focus of this work is the transformation between the hADL model and

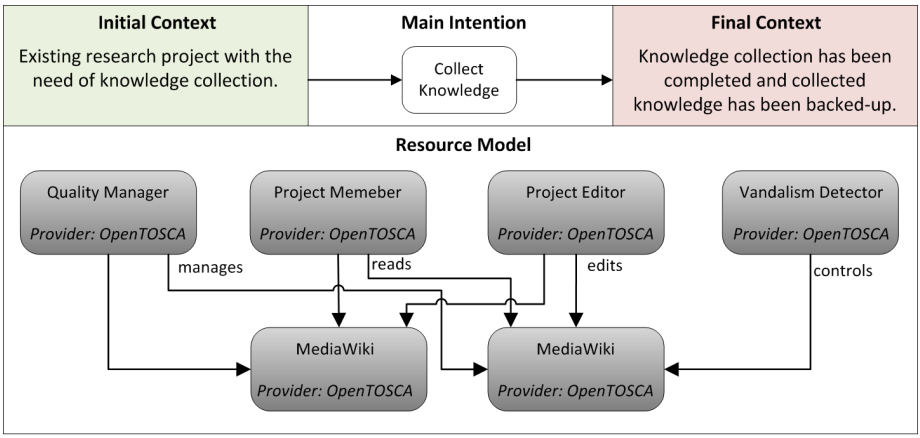

Fig. 2. Schematic IPE model of the dual wiki scenario

the resource model of an IPE model. In the following chapter, we will detail the interplay of hADL and IPE models and the relationships between them.

3 Approach

hADL models define high-level collaboration structures comprising human components, collaboration connectors, and collaboration objects without having to specify the underlying low-level deployment or implementation details. Instantiation of hADL models in an organizational context requires mapping hADL elements and their relations to specific, available resources of the respective organization. These resources abstract from provisioning details such as how a particular wiki is made available (e.g., in house on demand deployment via OpenTOSCA vs. external Software-as-a-Service provisioning) or how human participants become involved (e.g., via an organization internal OpenSocial-based platform vs. a LinkedIn external business social network). Resource relationships further specify which resources interact, but not how this interaction should be configured collaboration-wise. This concern, in turn, is best modeled in hADL.

Our approach (Fig. 3) aims at joining high-level hADL structures and low-level resources within the scope of an IPE model. Rather than directly mapping hADL and resource elements, our approach foresees their loose coupling via capabilities. hADL elements reference those capabilities which the ultimately selected resources needs to be able to fulfill. In contrast, resources specify both their complete capability set (e.g., for human-centric resources determined by their skill set), and their correspondingly required capabilities (to be provided by other resources). Our transformation logic takes a hADL model, capability definitions, and resources and produces an Informal Process Essentials (IPE) model. We outline the detailed transformation steps in the following section (Sec.4).

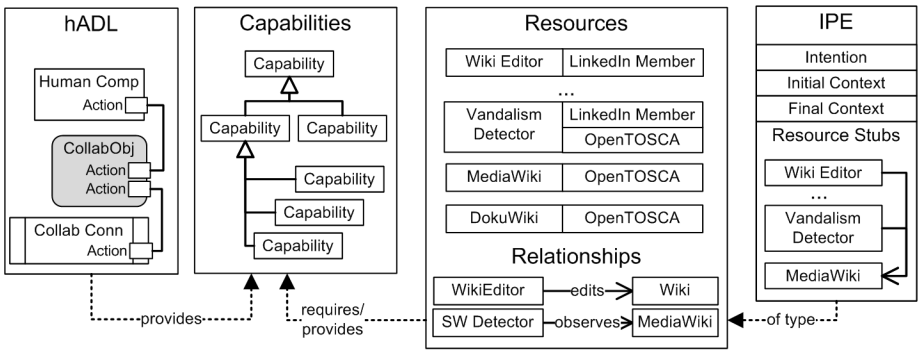

Fig. 3. Overview of models involved in deploying collaboration structures and their relations

Ultimately, the IPE model specifies a collaboration structure build from the participating resources, goals, and their initial/final contexts. It includes all details required for deploying and initializing resources upon informal process launch as well as their release at the end of the respective process. IPE models and corresponding tools (see Sec.5) thus constitute a realization environment for hADL models.

4 Transformation

The hADL-to-IPE transformation process relies on two complementary user roles. On the one hand, the *transformation principal* represents the informal process participant who drives the transformation for configuring the desired collaboration resources. In our scenario, one researcher from the lead project organization may assume this role. On the other hand, *resource principals* represent IT-level personnel who specify and maintain entries in the resource repository, capability repository, and hADL repository (see Fig. 5 center). Experts from an in-house IT department or at an external collaboration service provider typically assume this role.

4.1 Transformation Base Data

Our transformation assumes a bottom-up approach. Resource principals manage resource realizations. They utilize resource organizers such as OpenTOSCA for capturing the details required for instantiating and running resources (recall Section 2). Before a resource may become part of an IPE model, the resource principals first need to enhance resources with references to provided and required capabilities.

Resource principals also specify initial, basic hADL models; likewise with respective mappings to these capabilities. These basic hADL models primarily

determine the structure of collaboration objects including their available actions but not their configuration in an actual informal process. For instance, in the motivating scenario (Section 1.1), the External and Internal Wiki (part of the hADL model under design) are built on top of a Base Wiki structure (from the hADL model repository). With increased use, the repository may eventually contain complete, specific hADL models depending on how useful the participants found the deployed configuration upon informal process completion. The management procedures involving resource principals, however, are outside the scope of this paper and thus not further discussed.

The transformation process relies on the concept of shared capabilities to bridge the different levels of abstraction, i.e., hADL and IPE, respectively, resource models. Typically, resources cannot operate on their own but require additional capabilities (i.e., requirements) that are provided by other resources. *Resource relationships* capture resource pairs with matching provided/required capabilities. A *Uses Wiki* relationship, for example, binds a Wiki Editor resource (requiring a Wiki capability) to a Wiki resource (providing a Wiki capability). Resource relationships are first class model elements at the same level of resources and managed within the resource repository.

Similar to resources, hADL human components, collaboration connectors, and collaboration objects reference capabilities. To this end, the capability model (visualized in Fig. 4) defines a core capability hierarchy for further extensions according to HumanComponent capability, CollaborationConnector capability, and CollaborationObject capability. Root-level BaseCapabilityProperties and customization thereof specify further details (based on hADL actions) needed for collaboration-centric configuration of domain-specific resources. Specifically, the *BaseCapabilityProperties* element contains basic hADL actions and/or composite actions. Example action specification refinements thereof include a simple *Wiki permission* in the former case and a *Wiki permission group* in the latter case, both defined within *WikiPermissionProperties*. The hADL model utilizes these refinement specifications whenever a human component, collaboration connector, or collaboration object exposes the corresponding capability. A hADL collaboration object exposing a *WikiCapability*, for example, may then exhibit simple Wiki permission actions and Wiki permission group actions. Likewise, the scenario's *External Expert Editor* exposes the *WikiEditorCapability* capability which in turn relies on the *WikiCapability* (via resource relationships), and thus calls for action types from the WikiPermissionProperties.

4.2 Transformation Procedure

The hADL-to-IPE transformation process is semi-automatic. The transformation principal needs only become involved for resolving ambiguities, select among multiple choices, and confirm the final IPE configuration. Fig. 5 displays the ideal sequence of the various transformation activities, i.e., error handling is omitted for sake of clarity.

The transformation principal engages in the hADL-to-IPE transformation process by creating a hADL model. The principal selects suitable collaboration

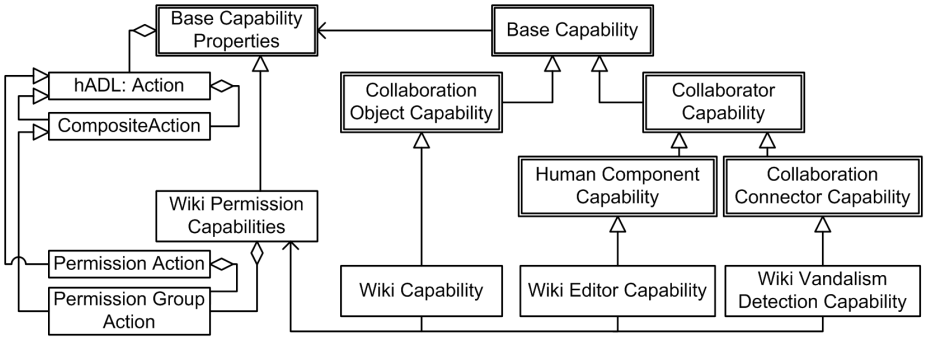

Fig. 4. Collaboration-centric capability model (double-edged elements are part of the base model, single-edged elements constitute extensions)

objects, human components, collaboration connectors, and structures thereof from the hADL repository. In our motivating scenario, the hADL repository contains the Wiki Base Structure and basic editor and manager components and connectors. These components and connectors typically exhibit only simple or even no actions but refer to capabilities. The transformation principals conducts three types of interleaving modeling activities.

- configuring the desired collaboration object configuration through selecting prepared hADL collaboration objects and (where applicable) rewiring their substructures.
- specifying the actual required component and connectors by copying, aggregating, creating or refining existing hADL collaborators from the hADL repository (e.g., merging a wiki reader with a notification dispatcher).
- linking collaborator actions and object actions, thereby completing the hADL model instance.

The resulting hADL model components, connectors, and objects exhibit a reference to their respective **provided** capabilities only. The required capabilities are automatically derived from the collaboration links during the *Generate IPE Stubs* step.

The IPE model determines a capability's configuration in terms of hADL actions. Whenever a collaboration object, component or connector contains a substructure, the specific wiring to subactions, therefore, needs pulling into the super structure. Otherwise, the information contained in the substructure mapping will be lost. The corresponding *Flatten hADL Model* step takes the initial hADL model and transforms it into a new hADL model where each action directly contains its related subactions.

The *Generate IPE Stubs* step creates a partial IPE model. The transformation logic generates an IPE *Resource* stub for each human component, collaboration connector, and collaboration object that references a (provided) capability. For instance, in the motivating scenario, the IPE model consists of stubs for

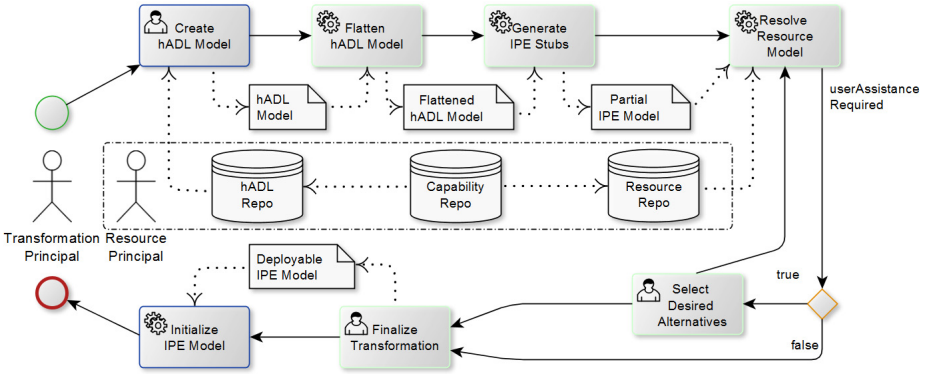

Fig. 5. hADL to IPE model transformation process (dotted lines represent dataflow, full lines depict control flow)

the External Editor, the Wiki Quality Manager, the External Wiki, the Internal Wiki, etc. A resource stub yet lacks a reference to an actual, existing resource enacting the respective hADL role in an informal process. Instead, the stub references the provided and required capabilities and provides their configuration in terms of the respective CapabilityProperties. The Project Editor resource stub, for example, features an IPE capability specification in the form of a WikiPermissionProperties element containing all hADL Project Editor actions (i.e., EditInternal, EditExternal) by copying them from the hADL model. A hADL link between a collaboration connector or a human component and a collaboration object implies that the former requires the capability of the latter. The links further define the precise subset of actions relevant to the requirement specification. Hence, the Project Editor stub features two IPE requirement specifications, both in the form of a WikiPermissionProperties element, each containing the linked hADL actions of the two hADL wiki objects (Internal Wiki - Editing composite action and External Wiki - Editing composite action).

The transformation logic additionally generates IPE *Resource Relationship* stubs based on the links in the flattened hADL input model. Resource relationship stubs describe which resource (stub) requires what other resource(s) (stubs) within the scope of the IPE model instance. The resource relationships ensure that ultimately the deployed Project Member resources, Project Editor resources, and Wiki Quality Manager resources have access to the same Internal Wiki resource, rather than each one having access to three separate wikis merely configured to each individual user resource. The exact relationship type, however, remains undecided as long as the IPE resource stubs remain without assigned resources from the repository. The concrete resource relationship type between Vandalism Detector and External Wiki depends on whether the ultimate resource will be software-based (and thus requires access to the Wiki via a machine API) or human-based (and thus requires access to the Wiki via a human consumable Web interface).

The *Resolve Resource Model* step aims to automatically complete the partial IPE model. The step retrieves all resources from the resource repository that match the given capability provisioning and capability requirement specification. The capability matching algorithm produces a list of candidate resources for each resource stub. The number of candidates determines the subsequent options:

- No candidates: When no resource provides a particular capability the transformation principal needs to go back to the initial hADL modeling step and decide whether to remove the respective hADL collaboration element, redefine its capabilities, or inform a resource principal to add new or enhance existing resources in the repository to allow for successful matching.
- Exactly one candidate: The ideal case as no transformation principal intervention is required.
- Multiple candidates: The transformation principal needs to *Select Desired Alternatives* upon which the control flow returns to the *Resolve Resource Model* step. For instance, a *VandalismDetectionCapability* can be provided by two different resources, e.g., a human resource or a software analyzer which detects violation patterns and notifies a users.

Having resolved the IPE resource stubs to concrete resources from the repository, the *Resolve Resource model* step commences with resolving the relationships stubs. During this process, the matching algorithm iterates through all relationship stubs and extracts compatible concrete relationship specifications candidates from the resource repository. To this end, the algorithm extracts the concrete resources references in the relationship stub and analyzes which relationship specification binds such a resource pair.

Similarly to before, having multiple relationship candidates requires the intervention of the transformation principal. Note that having no candidate for a particular resource relationship stub doesn't necessarily imply the need for restructuring the hADL model or resource relation specifications but rather having selected incompatible resources. For example, resource relationships in the repository may specify that a human Vandalism Detector is able to use both MediaWiki and DokuWiki resources while a software-based Detector is limited to MediaWiki. Hence, when selecting earlier the software-based Detector resource (for the VandalismDetection capability) and the DokuWiki resource (for the WikiCapability), the algorithm cannot find a concrete relationship. Consequently, the transformation process loops between *Resolve Resource Model* and *Select Desired Alternatives* until either all resource and resource relation stubs are resolved, or the resource principal aborts the process due to missing or mismatching resources and resource relationships, respectively.

4.3 Transformation Output Application

The hADL models lack information on the intentions and initial/final context of informal processes. The transformation principal completes the IPE model with these missing details in the *Finalize Transformation* step. This step also gives

the resource principal a last opportunity for checking the generated resource model for completeness and correctness and adjusting it as needed. After these final manual configurations, the IPE model is ready for deployment whenever the execution environment matches the initial process context. For deployment, the various resource organizers obtain the IPE model, extract the hADL actions from each resource's capability requirements and capability provisioning specification for configuring and initializing the actual resource instances. For example, the WikiPermission actions and WikiPermissionGroup actions determine directly MediaWiki's permission configuration file.

5 Case Study Based on the OpenTOSCA Ecosystem

We have created a prototype for transformation procedure (all steps except for the first and last in Fig. 5) on top of our previous work [30] in order to test and validate our approach (see screenshots in Fig. 6). We realized the prototype as a REST-based web-service that reuses following specifications and tools: We apply the Topology Orchestration Specification for Cloud Applications (TOSCA) [6] for resource deployment. OpenTOSCA [5] is an open-source container (i.e., resource organizer) for cloud applications defined in TOSCA. Winery [24] is the corresponding modeling tool for specifying TOSCA-based applications. As TOSCA supports the concepts of capability and requirements, we use Winery as repository for capabilities, resources, and resource relationships. We defined domain-specific capabilities (in the *wikiCap* namespace) and corresponding capability property and action specification (in the *wtypes* namespace) for implementing the case study based on the motivating scenario. Due to page constraints we only provide some model excerpts in this section. A complete set of XML schemas, specifications, XSL transformations, input and output models as well as a proof-of-concept tool is available as supporting online material (SOM) at http://co-act.biz/downloads.

First, the transformation principal creates the hADL model using a hADL editor (outside of this paper's scope). The hADL editor loads the predefined, shared capabilities from Winery. The principal annotates the hADL modeling elements with the desired capability definitions (in our case those defined in the *wikiCap* namespace). The Listing 1.1 presents the hADL XML excerpt for the External Expert Editor from the motivating scenario Section 1.1. hADL's extension mechanism enables annotating each human component, collaboration connector, and collaboration object with the CapabilityRef elements containing the qualified name of a capability, e.g., *wikiCap:WikiEditorCapability* (see Listing 1.1 line 3).

```
<hADL:name>External Expert Editor</hADL:name>
  <hADL:extension>
    <depl:CapabilityRef>wikiCap:WikiEditorCapability</depl:CapabilityRef>
  </hADL:extension>
  <hADL:action id="external-expert-editExt">
    <hADL:name>edit</hADL:name>
    <hADL:primitive>CREATE</hADL:primitive>
    <hADL:primitive>READ</hADL:primitive>
    <hADL:primitive>UPDATE</hADL:primitive>
  </hADL:action>
</hADL:component>
```

Listing 1.1. hADL Human Component, XML excerpt

Running the set of XSL transformation and resource resolving steps (see upper right in screenshot Fig. 6) results in the IPE model (excerpt) in Listing 1.2 (full output available as SOM). The hADL component in Listing 1.1 becomes an IPE resource (line 1) consisting of a single provided capability (line 5 to 15), single requirement (line 17 to 37), and single relationship (line 39 to 45).

- the IPE capability segment simply refers to the provided capability (line 6) and a copy of the only Extern Expert's action defined in the hADL model.
- The IPE requirement segment references the required Wiki capability (line 17) as derived through navigating the hADL links. The requirement additionally exhibits a copy of the linked external wiki's expert editing action (lines 21 to 35) of type *WikiPermissionSetAction*. Note the action's domain specific element *groupName* (line 35) and subactions (line 28 to 33) used to configure the actual Wiki resource instance upon deployment.
- The IPE resource relationship segment indicates that the IPE resource External Editor ties to the IPE resource VWiki-Complex (line 43).

The IPE model content explained up to here represents the stub information and needs completion with actual resources from resource repository. In our case study, we search through human resources and IT resources in Winery. We converted also all available human resources to TOSCA NodeTypes and stored them in Winery. We subsequently iterate through all available human and IT resources. When checking a resource, we need to ensure that not only all hADL derived capability requirements are satisfied but also those referenced by the resources in Winery. At last, the relationships are selected based on the selected resources. In Listing 1.2, the stubs become complete by adding following resource, respectively relationship reference:

- External Editor specification maps to resource *wikiRes:Editor* (line 4).
- Relationship to VWiki-Complex maps to resource relationship *wikiRes-Rel:edits* (line 40).

The resulting IPE Resource Model (see screenshot in Fig. 6 lower left) provides all necessary details to realize the desired resource model. In this case, it references certain TOSCA NodeTypes and RelationshipTypes. During initialization of the respective model, the resource organizer receives the resource information and executes the necessary domain specific operations. In case of TOSCA

resources, first a topology is generated by converting TOSCA types (i.e., NodeTypes and RelationshipTypes) to TOSCA templates (i.e., NodeTemplates and RelationshipTemplates). Work of Hirmer et al. [22] supports the generation of a complete topology. Finally, we end up with a service template which contains the application topology. Such service templates can be deployed on declarative TOSCA containers [12].

```
<ipsm:Resource ipsm:id="MediaWiki.ExternalExpert" ipsm:name="External Expert Editor"
  ipsm:realizationDomain="http://www.uni-stuttgart.de/opentosca"
  xmlns:wikiCap="http://www.iaas.uni-stuttgart.de/ipsm/hadl/case-study/tosca/types"
  ipsm:type="wikiRes:Editor">
  <ipsm:CapabilityList>
    <ipsm:Capability ipsm:type="wikiCap:WikiEditorCapability">
      <ipsm:PropertyList>
       <hADL:action xmlns:hADL="http://at.ac.tuwien.dsg/hADL/hADLcore"
                    id="external-expert-editExt">
         <hADL:name>edit</hADL:name>
         <hADL:primitive>CREATE</hADL:primitive>
         <hADL:primitive>READ</hADL:primitive>
         <hADL:primitive>UPDATE</hADL:primitive>
       </hADL:action>
    ...
  <ipsm:RequirementList>
   <ipsm:Requirement  ipsm:requiredCapability="wikiCap:WikiCapability">
     <ipsm:PropertyList>
       <hADL:action
          xmlns:hADL="http://at.ac.tuwien.dsg/hADL/hADLcore"
          xmlns:wtypes="http://www.iaas.uni-stuttgart.de/ipsm/hadl/case-study/types"
          id="virtualwiki2-expert"
          xsi:type="wtypes:tMediaWikiPermissionSetAction">
         <hADL:name>Expert Editing</hADL:name>
         <hADL:primitive>READ</hADL:primitive>
         <hADL:primitive>UPDATE</hADL:primitive>
         <hADL:primitive>DELETE</hADL:primitive>
         <!-- subactions from references hADL substructure -->
         <btypes:SubAction
         xmlns:btypes="http://www.iaas.uni-stuttgart.de/ipsm/hadl/base/types"
             id="page-read" xsi:type="wiki:tMediaWikiPermissionAction">
            <hADL:name>read</hADL:name>
            <wiki:permission>read</wiki:permission>
         </btypes:SubAction>
         <!-- further subactions -->
         <wtypes:groupName>expert</wtypes:groupName>
       </hADL:action>
    ...
  <ipsm:RelationshipList>
    <ipsm:Relationship  ipsm:sourceDomain="http://www.uni-stuttgart.de/opentosca"
                        ipsm:type="wikiResRel:edits">
      <ipsm:TargetResourceList>
        <ipsm:TargetResource ipsm:targetDomain="http://www.uni-stuttgart.de/opentosca">
           VWiki-Complex</ipsm:TargetResource>
      </ipsm:TargetResourceList>
    </ipsm:Relationship>
  ...
```

Listing 1.2. Human Component as an IPE Resource, XML excerpt

Discussion. Our case study demonstrates the feasibility of separating collaboration-level resource configuration and system-level resource deployment of a real-world collaboration tool. Transformation principals need not know any technical details of the underlying technical infrastructure (here the configura-

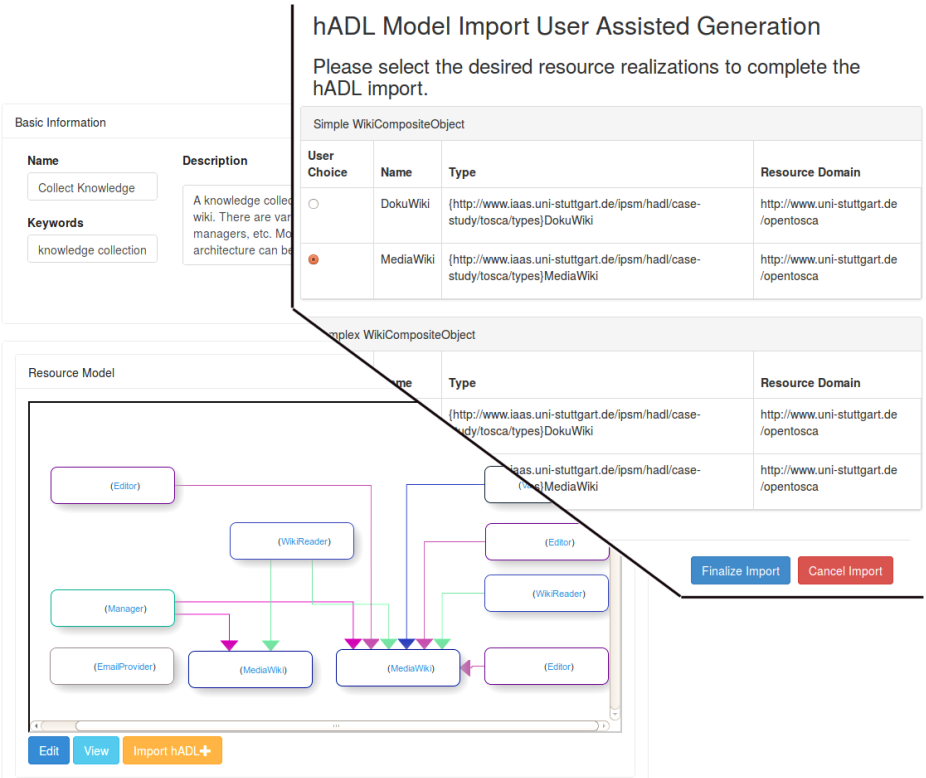

Fig. 6. hADL-to-IPE prototype screenshots

tion of MediaWiki software, web server, database, and hosting environment). The hADL model enables the principal to quickly perceive which permissions affect what collaboration mechanism (e.g., page, file, history) of a MediaWiki installation. Having multiple informal process participants access two wikis with different capabilities exemplifies our framework's ability to transform non-trivial, real-world hADL models to IPE models. We believe it's in the interest of the reader to keep subsequent deployment details out of this paper's scope as we build our case study on top of proven and mature tools and standards such as the OpenTOSCA container and Winery.

Our approach is applicable to collaboration platforms beyond wikis. The teaching support platform Moodle, for example, exhibits extensive configurability. Similar to MediaWiki, it supports the aggregation of collaboration capabilities into custom roles[3] and hence would be directly applicable to modeling in hADL and subsequent transformation to IPE.

[3] https://docs.moodle.org/24/en/Creating_custom_roles

6 Related Work

Web-based platforms, languages, and specifications such as CrowdSearcher [8], Jabberwocky [4], CrowdForge [23], or CrowdLang [25] aim at efficiently executing large-scale human-centric workflows. These approaches focus primarily on achieving sufficient quality at low costs when distributing, collecting, and filtering massive amounts of tasks. Interaction among task workers is not foreseen; all dependencies are modeled as task-centric workflow patterns [9]. Brambilla and Mauri integrate social network-centric actions into web applications via social primitives [11]. Their focus is on public social platforms (Facebook, Twitter, ...) and thus configuration and deployment of the collaboration structures remains out of scope.

Recently research efforts started explicitly targeting the integration of social media into business process management (BPM) technology. Brambilla et al. present design patterns for integrating of social network features in BPMN [10]. A social network user may engage in task-centric actions such as voting, commenting, reading a message, or joining a task. Böhringer utilizes tagging, activity streams, and micro-blogging for merging ad-hoc activities into case management [7]. Dengler et al. utilize collaborative software such as Wikis and social networks for coordinating process activities [14]. oBPM [20] is an approach for opportunistically modeling business processes in a bottom up manner. It thereby relies on task and artifact abstraction for coordination among participants. These approaches differ in two crucial aspects from our work: (i) they rely on a predefined process model, and (ii) they statically integrate social media resources.

The BPM community recognized the need for flexible processes early on [28], distinguishing among flexibility *by design*, *by deviation*, *by underspecification*, and *by change* [29]. Work on process flexibility, however, has the primary focus on the process specification and not on how to enable executable support for collaboration and coordination among process participants. Even traditional workflow description languages dedicated to modeling the human involvement such as Little-JIL [13], BPEL4People [1], or WS-HumanTask [3] foresee no explicit communication among process participants outside of tasks. Although BPEL4people supports four eyes, nomination, escalation, and chained execution scenarios; and WS-HumanTask allows attaching comments to tasks, all interaction is purely task-centric.

In our own recent work [15] we explored the integration of business process and collaboration patterns but didn't address ad-hoc collaboration resource deployment nor the context of informal processes. These informal processes are addressed by different approaches such as adaptive case management [21] or activity-centric computing [18]. However, these approaches focus primarily on activities and not resources. To this end, we have proposed a resource-oriented approach [30]. It enables the deployment of the resources in the context of an informal process context.

7 Conclusion and Outlook

Traditional workflow and activity-centric approaches are inadequate when organizations need to document and reuse best practices for solving problems of collaborative nature. Instead, tacit knowledge on suitable collaboration structures and their enactable informal process models constitute more fitting concepts. To this end, we presented a novel approach for transforming collaboration-level models (applying the human Architecture Description Language) into deployable technical informal processes (represented in IPE). We introduced the concept of shared capabilities as basis for transformation across different levels of abstraction. We subsequently detailed the various transformation steps and described their application in the scope of a validating case study and proof-of-concept tool involving real-world IT resources (i.e., MediaWiki) and human resources.

Having focused on the technical aspects (models, transformation, tools) in this paper, we are planning for more detailed, validating experiments involving more complex scenarios that we couldn't outline here due to page restrictions. These experiments will provide quantitative measures that serve as evidence on the benefits of our approach.

Acknowledgments. This work has been partially supported by Graduate School of Excellence advanced Manufacturing Engineering (GSaME)[4] and by the EU FP7 SmartSociety project, under Grant No. 600854.

References

1. BPEL4People. http://docs.oasis-open.org/bpel4people/bpel4people-1.1.pdf
2. BPMN 2.0. http://www.omg.org/spec/BPMN/2.0/PDF/
3. WS-HumanTask. http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cs-01.pdf
4. Ahmad, S., Battle, A., Malkani, Z., Kamvar, S.: The jabberwocky programming environment for structured social computing. In: UIST 2011, New York, NY, USA, pp. 53–64 (2011)
5. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA – a runtime for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 692–695. Springer, Heidelberg (2013)
6. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: TOSCA: Portable Automated Deployment and Management of Cloud Applications. In: Bouguettaya, A., Sheng, Q.Z., Daniel, F. (eds.) Advanced Web Services, pp. 527–549. Springer, New York (2014)
7. Böhringer, M.: Emergent case management for ad-hoc processes: a solution based on microblogging and activity streams. In: Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 384–395. Springer, Heidelberg (2011)
8. Bozzon, A., Brambilla, M., Ceri, S., Mauri, A.: Reactive crowdsourcing. In: WWW 2013, pp. 153–164. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2013)

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

9. Bozzon, A., Brambilla, M., Ceri, S., Mauri, A., Volonterio, R.: Pattern-based specification of crowdsourcing applications. In: Casteleyn, S., Rossi, G., Winckler, M. (eds.) ICWE 2014. LNCS, vol. 8541, pp. 218–235. Springer, Heidelberg (2014)
10. Brambilla, M., Fraternali, P., Vaca, C.: BPMN and design patterns for engineering social BPM solutions. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 219–230. Springer, Heidelberg (2012)
11. Brambilla, M., Mauri, A.: Model-driven development of social network enabled applications with WebML and social primitives. In: Grossniklaus, M., Wimmer, M. (eds.) ICWE Workshops 2012. LNCS, vol. 7703, pp. 41–55. Springer, Heidelberg (2012)
12. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining declarative and imperative cloud application provisioning based on TOSCA. In: Proceedings of the IEEE International Conference on Cloud Engineering, pp. 87–96 (2014)
13. Cass, A.G., Lerner, B.S., Sutton Jr. S.M., McCall, E.K., Wise, A.E., Osterweil, L.J.: Little-JIL/Juliette: a process definition language and interpreter. In: ICSE 2000, pp. 754–757. IEEE (2000)
14. Dengler, F., Koschmider, A., Oberweis, A., Zhang, H.: Social software for coordination of collaborative process activities. In: Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 396–407. Springer, Heidelberg (2011)
15. Dorn, C., Dustdar, S., Osterweil, L.J.: Specifying flexible human behavior in interaction-intensive process environments. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 366–373. Springer, Heidelberg (2014)
16. Dorn, C., Taylor, R.N.: Architecture-driven modeling of adaptive collaboration structures in large-scale social web applications. In: Wang, X.S., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 143–156. Springer, Heidelberg (2012)
17. Dorn, C., Taylor, R.N.: Analyzing runtime adaptability of collaboration patterns. Concurrency Computat.: Pract. Exper. (2014)
18. Dustdar, S.: Caramba Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams. Distributed Parallel Databases **15**, 45–66 (2004)
19. Ellis, C., Nutt, G.J.: Workflow: the process spectrum. In: Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems, pp. 140–145 (1996)
20. Grünert, D., Brucker-Kley, E., Keller, T.: oBPM – an opportunistic approach to business process modeling and execution. In: Fournier, F., Mendling, J. (eds.) BPM 2014 Workshops. LNBIP, vol. 202, pp. 463–474. Springer, Heidelberg (2015)
21. Herrmann, C., Kurz, M.: Adaptive case management: supporting knowledge intensive processes with IT systems. In: Schmidt, W. (ed.) S-BPM ONE 2011. CCIS, vol. 213, pp. 80–97. Springer, Heidelberg (2011)
22. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F.: Automatic topology completion of TOSCA-based cloud applications. In: Proceedings of CloudCycle14 Workshops, pp. 247–258. Bonn (2014)
23. Kittur, A., Smus, B., Khamkar, S., Kraut, R.E.: CrowdForge: crowdsourcing complex work. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST 2011, pp. 43–52. ACM, New York (2011)
24. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – a modeling tool for TOSCA-based cloud applications. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 700–704. Springer, Heidelberg (2013)

25. Minder, P., Bernstein, A.: CrowdLang - first steps towards programmable human computers for general computation. In: AAAI Workshops (2011)
26. BPEL 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf
27. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
28. Sadiq, S.K., Sadiq, W., Orlowska, M.E.: Pockets of flexibility in workflow specification. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 513–526. Springer, Heidelberg (2001)
29. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., Aalst, W.: Process flexibility: a survey of contemporary approaches. In: Dietz, J.L.G., Albani, A., Barjis, J. (eds.) CAiSE 2008. LNBIP, vol. 10, pp. 16–30. Springer, Berlin Heidelberg (2008)
30. Sungur, C.T., Binz, T., Breitenbücher, U., Leymann, F.: Informal process essentials. In: EDOC 2014, pp. 200–209. IEEE (2014)