World Wide Web
DOI 10.1007/s11280-012-0180-6

Crowdsourcing tasks to social networks in BPEL4People

Daniel Schall · Benjamin Satzger · Harald Psaier

Received: 24 November 2011 / Revised: 28 May 2012 / Accepted: 19 July 2012

Abstract Human-interactions are a substantial part of today's business processes. In service-oriented systems this has led to specifications such as WS-HumanTask and BPEL4People which aim at standardizing the interaction protocol between software processes and humans. These specifications received considerable attention from major industry players due to their extensibility and interoperability. Recently, crowdsourcing has emerged as a new paradigm for leveraging a human workforce using Web technologies. We argue that crowdsourcing techniques and platforms could benefit from XML-based standards such as WS-HumanTask and BPEL4People as these specifications allow for extensibility and cross-platform operation. However, most efforts to model human interactions using BPEL4People focus on relatively static role models for selecting the right person to interact with. Thus, BPEL4People is not well suited for specifying and executing processes involving crowdsourcing of tasks to online communities. Here, we extend BPEL4People with non-functional properties that allow to cope with the inherent dynamics of crowdsourcing processes.

D. Schall (✉)
Siemens Corporate Technology (CT T CEE), Siemensstrasse 90, 1211 Wien, Austria
e-mail: daniel.schall@siemens.com

B. Satzger · H. Psaier
Distributed Systems Group, Vienna University of Technology,
Argentinierstrasse 8, 1040 Wien, Austria

B. Satzger
e-mail: Satzger@infosys.tuwien.ac.at

H. Psaier
e-mail: Psaier@infosys.tuwien.ac.at

Such properties include human capabilities and the level of skills. We discuss the formation of social networks that are particularly beneficial for processing extended BPEL4People tasks. Furthermore, we present novel approaches for the automated assignment of tasks to a social group. The feasibility of our approach is shown through a proof of concept implementation of various concepts as well as simulations and experiments to evaluate our ranking and selection approach.

Keywords crowdsourcing · BPEL4People · non-functional properties · social networks

1 Introduction

Web services have paved the way for a new type of distributed system. Services let developers and engineers design systems in a modular manner, adhering to standardized interfaces. Services already play an important role in fulfilling organizations' business objectives because process stakeholders can design, implement, compose, and execute business processes using Web services as well as languages such as the Business Process Execution Language (BPEL) [5].

However, the BPEL specification was lacking a concept of (process) activities that are performed by human actors. Specifically the case that certain services in a process need to be provided by people is not covered. Recently, major software vendors have been working on standards addressing the lack of human interaction support in service-oriented systems. WS-HumanTask [4] (WS-HT) and BPEL4People [3] (B4P) were released to address the emergent need for human interactions in business-oriented processes. These standards specify languages for modeling human interactions, the lifecycle of humans tasks, and generic role models.

Meanwhile, a new Web-based business model called *crowdsourcing* attempts to harnesses the creative solutions of a distributed network of individuals established with the goal to *outsource* tasks to these workers [9, 20, 53]. This network of humans is typically an open Internet-based platform that follows the open world assumption and tries to attract members with different knowledge and interests. Large IT companies such as Amazon, Google, or Yahoo! have recognized the opportunities behind such mass collaboration systems [14] for both improving their own services and as business case. While the aforementioned specifications have been created to model human interactions in BPEL, it remains an open issue how to apply them to crowdsourcing. Specifically, the WS-HT specification does not define any particular mechanisms to find or select people in open and dynamic environments. Instead, a *Logical People Group* is used to query an organizational directory. We believe that human interactions in SOA need to be supported in a flexible manner, in particular, it should be possible to use crowdsoucing for process execution.

In this work we present the following key contributions:

1. *An approach* for combining crowdsourcing techniques and B4P related XML standards.
2. *B4P with non-functional properties* for adaptive and quality-aware crowdsourcing of service-oriented processes.
3. *Social community formation* for efficient crowdsourcing of processes.

4. *Automated matching* of tasks to members of a social crowdsourcing community.
5. *Implementation and evaluation* of our concepts.

This paper is structured as follows: In Section 2 we discuss related work, in Section 3 we present the application areas of our proposed service-oriented crowdsourcing environment and outline the steps of our approach. In Section 4 we propose extensions to introduce non-functional requirements for B4P. In Section 5 we explain the establishment of a social network structure that allows for an efficient execution of B4P processes. We present the automated matching of extended B4P processes to a social community crowd in Section 6. Finally, we present in Section 7 the results of our evaluation and conclude our paper in Section 8.

2 Related work

The work presented in this paper focuses on a methodology and tools allowing human interactions in SOA to be executed in a flexible manner. We structure our discussion of related work in the area of crowdsourcing, SOA, Web services, and process-centric collaboration. Second, we present related research in adaptive systems engineering. Third, we discuss approaches and metrics for network analysis.

In recent years, there has been a growing interest in the complex 'connectedness' of today's society. Phenomena in our online-society involve networks, incentives, and the aggregate behavior of groups [17]. Peer-production distinguishes from the property- and contract-based models of firms and markets. Research by [8] analyzes how groups of individuals successfully collaborate on large-scale projects following a diverse cluster of motivational drives and social signals. In contrast, *Human computation* is motivated by the need to outsource certain steps in a computational process to humans [18, 42]. An application of human computation in genetic algorithms was presented in [25]. A variant of human computation called 'games that matter' was introduced by [52]. Related to human computation are systems such as Amazon Mechanical Turk[1] (MTurk). MTurk is a Web-based, task-centric platform in which users can publish, claim, and process tasks. [50] evaluates the task properties of a similar platform in cases where large amounts of data are reviewed by humans. In contrast to common question/answer (Q/A) forums, such as Yahoo! Answers,[2] MTurk enables businesses to access the manpower of thousands of people using a Web services API. Mixed service-oriented systems [41] and [42] target flexible interactions and compositions of *Human-Provided Services (HPS)* [45] and *Software-Based Services (SBS)*. This approach is aligned with the vision of the Web 2.0, where people can actively contribute services. In such networks, humans may participate and provide services in a uniform way by using the HPS framework [41]. A similar vision is shared by [36] defining *emergent collectives* which are networks of interlinked valued nodes (services). In such collectives, there is an easy way to add nodes by distributed actors so that the network will scale. Current crowdsourcing platforms offer very limited

[1] http://www.mturk.com/

[2] http://answers.yahoo.com/

support for modeling complex interactions (e.g., sub-task processing) that require coordination of humans' joint capabilities and software-based services.

Several trends originated from human interactions in service-oriented systems. As mentioned before, B4P defines human interactions in business processes via the human task specification [4]. A concrete implementation of B4P as a service has been introduced in [51], but without supporting process adaptivity. Worklets [1] grounded in *activity theory* represent self-contained subprocesses. Another approach for flexible activities in business-oriented environments was presented in [12]. In [39], the relation of various B4P-related Web standards and resource patterns is discussed. In contrast to process-centric compositions in SOA, task-based crowdsourcing platform such as MTurk do not support long-running interactions and compositions of humans and services. The problem of composition is strongly related to organization and control. The key principles of autonomic computing (e.g., see [21] for an overview) aim at supporting systems featuring *self-** properties (e.g., failure resilience through self-organizing computing elements). As an example, Maximilien and Singh [29] focus on autonomic services and trusted service selection. In [28], a reference architecture for self-organizing service-oriented systems is presented, but without considering humans 'as part' of the system. The authors in [19] propose adaptive flows to support flexibility and evolution in collaborative, pervasive environments. Technological and social aspects shape the operation constraints of a system [24]. It is, therefore, important not only to model human interactions in process-centric systems, but also to understand how people are connected [47, 54] and how information flows are influenced by structure. When building Web-centric applications involving human tasks, one has to consider incentive schemes that are likely to encourage users to perform these tasks that crucially rely on human input [48]. In this work we show how human metrics can be used in adaptive processes.

Human tasks metrics in workflow management system have been discussed in [26]. A formal approach to modeling and measuring inconsistencies and deviations, generalized for human-centered systems, was presented in [13]. In [56], flexible process views in Web service environments are proposed to cater for the diverse interests, authority levels, etc., of different users. Studies on distributed teams focus on human performance and interactions [6, 35], as well as in *Enterprise 2.0* environments [11]. Models and algorithms to determine the expertise of users are important in future service-oriented environments [43]. Task-based platforms allow users to share their expertise [54]; or users offer their expertise by helping other users in forums or answer communities [2]. By analyzing email conversations [16], the authors studied graph-based algorithms such as Hyperlink-Induced Topic Search (HITS) [23] and PageRank [34] to estimate the expertise of users. The authors in [46] used a graph-entropy model to measure the importance of users. The work by [55] applied HITS as well as PageRank in online communities (i.e., a Java question and answer forum). However, existing approaches for expertise mining have not been applied in service-oriented crowdsourcing environments.

In our previous work, we have designed and implemented the HPS framework [41, 42, 45] allowing users to define services and to provision human expertise in a service-oriented manner. Furthermore, we have designed a market-based crowdsourcing platform [40] and simulation environment to stimulate the evolution of user skills. Here we extend B4P and related XML standards to cope with the inherent dynamics of crowdsourcing environments. Also, we show how social networks can help to process crowdsourced tasks in a more efficient manner.

3 Service-oriented crowdsourcing

More than ever, business processes are affected by rapidly changing requirements and imperative adaptations that come along with necessary modernizations of the in-house activities and adjustments to the market. Many of today's workflow-based systems are still based on a *top-down* design for processes. It is clear, that there is a trend to the combination of interactions between humans and software based applications, such as SBS, as a central requirement in business environments. This may work fairly well for processes involving only SBS with minor human interaction. However, once the human interaction models in those processes become more important and complex, a top-down approach is unable to foresee and cope with the implications of the human behavior related dynamics. There are several types of tasks that are still best processed by humans.

3.1 Task-based crowdsourcing markets

Currently, the extensions to the specifications for business processes are designed for simpler human tasks, e.g., making process progress decisions and process approval requests [3]. Nevertheless, with the new marketplaces provided by crowdsourcing including workers with manifold skills, new types of tasks can be considered for outsourcing. Recently, many platforms have started to offer a versatile number of tasks that can be outsourced to the crowd.

In the following, we overview some of the potential crowd tasks that could be designed and outsourced using our approach:

– *Classification* or *categorization* tasks using for example the MTurk marketplace [22], CrowdFlower,[3] or SmartSheet.[4] Categorization is one of the most common use cases for crowdsourcing. A categorization task is one that asks a worker to select from a list of options.
– *Transcription* tasks as offered by CastingWords[5] or SpeechInk.[6] Transcription services include transcription of audio-to-text.
– *Web development* and web programming as provided by, for example, oDesk.[7] Web development tasks include integration of scripts with Web service APIs or programming questions regarding different frameworks or toolkits.

We foresee that our B4P-based approach can be used in scenarios like document translation, proof-reading and correction of documents (see also crowd-powered word processors[8]), transcription, data-cleansing, and simple programming tasks as those offered by oDesk. These tasks can be crowdsourced by creating appropriate B4P activities and WS-Human Tasks embeddings that are transmitted to the HPS

[3]http://www.crowdflower.com
[4]http://www.smartsheet.com
[5]http://www.castingwords.com/
[6]http://www.sppechink.com/
[7]http://www.odesk.com
[8]http://projects.csail.mit.edu/soylent/

middleware. The HPS middleware implements algorithms for matching, ranking, selection and runtime monitoring of tasks.

3.2 Approach outline

Here, we propose adaptive human interaction support in service-oriented systems. Human interaction support in SOA has only recently been proposed and supported by standards such as WS-HT [4] and B4P [3]. We argue that these standards need to be extended to support compositions of both SBS and HPS using crowdsourcing. The benefit of this approach is a seamless QoS-based service-oriented infrastructure that is able to adjust its interactions based on service-level agreements (SLAs) and quality constraints. Our previous work [40, 42] has already detailed the challenges of integrating in-house processes to a crowdsourcing environment. In this work, the approach is extended by support for crowdsourcing of composed, complex tasks. We have identified the following main challenges:

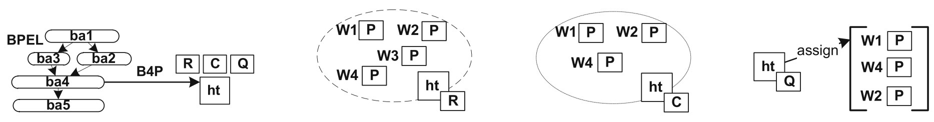
- *Crowd structure:* Composed tasks not only require humans for processing the set of subtasks, but also, a coordinated and supervised assignment and merging of the individual results to a final result. For this purpose, in this work we identify three roles. The *Coordinator*, on the one side, keeps in touch with the business process management, and, on the other side, maintains her/his relations to various crowd communities. The motivation of coordinators is to some extend similar to the role of a moderator in social-online communities such as *Slashdot*. Moderators, in our case coordinators, can be understood as gatekeepers [27] who control the quality of postings (in our case tasks) in online communities. The *Supervisor* represents a community and is aware of the current possible segmentation of one crowd task to a set of subtasks with the related distribution to her/his team. Finally, there are the common crowd *Workers*. We believe that this simple role model consisting of three distinct roles is sufficient for crowd tasks as described in Section 3.1. Crowd members temporarily form teams to work jointly on tasks on a magnitude of hours and days. Shortly after that the team dissolves again and workers pick some other task. More complicated hierarchies as found in enterprises or large-scale development teams may not be suitable for such short-lived groups.
- *Non-functional properties:* Current service-based process definitions for human interactions, in particular the combination of BPEL, B4P, and WS-HT definitions, need to be extended for the requirements of crowdsourcing. A further challenge addressed with crowdsourcing, is the integration of SOA agreements' specifications WSLA (Web Service Level Agreements)[9] in the outsourcing process.
- *Crowd member ranking:* Crowd environments are dynamic by nature. Therefore, it is vital to the outsourcing party that the current best matching crowd members can be detected and ranked according to the task's requirements. The result allows the customer to select from a large set of potential workers. Also, the final decision remains with the customer that can hide possibly sensitive selection constraints from the public crowd platform.

[9]http://www.research.ibm.com/wsla/WSLA093.xsd

An important aspect of this work is to introduce the notion of *expertise* (i.e., human skills) in the context of B4P. Existing approaches for expertise mining (e.g., see [55]) have mostly been applied in online communities or social network analysis, but not in process-oriented crowdsourcing environments. Here we introduce the notion of *capabilities* and *skills* in B4P to ensure quality-aware crowdsourcing of human tasks. In the following, we define important concepts used in this work:

- *Skills* are specific to the functions workers perform doing their job. As an example, a worker may perform activities related to a software development task such as reviewing code. The worker may be an expert in 'Java programming', a beginner in 'Python programming' and so forth. However, skills—as used in this work—are always based on *personal expertise* workers have and workers may improve their skills through training (e.g., improving the skill level from 'knowledgeable' to 'expert').
- *Capabilities* describe non-functional human properties to determine a workers suitable to work on a task. Human capabilities describe behavior properties which cannot be directly derived from the worker's profile. Example capabilities include 'worker should be capable of coordinating 5 other team members' or 'worker should be capable of merging and finalizing translated input from other workers'. Thus, the suitability of a worker is highly dependent on the current load conditions within the crowdsourcing environment. A worker might have in principle the skills to finalize a translated document but may not have the resources to merge the input from other workers. At B4P level, only capabilities are defined such as 'translate document and split/merge sections'. To fulfill this capability, workers are matched and ranked based on their skills, their social network connectivity (being able to split, distribute portions of the document to peers in the social network and to merge the received input) and their current load (number of pending tasks in a worker's queue).
- *Constraints* allow the customer and owner of the WS-HT to state some strict or relaxed filter options on the different roles. For example, from the customer point of view, certain crowd members may belong to a group *Preferred Workers* that is either populated automatically based on past experiences (e.g., reliable and trustworthy behavior of a worker towards the customer) or other customer internal policies. Constraints could state that only *Preferred Workers* should be able to review certain portions of a document. Other constrains could for example state that only workers from a certain geographical location may work on a task.

Figure 1 outlines the idea of the approach. In the first step in Figure 1a, part of a BPEL process includes a B4P extended activity (ba4) to transfer a set of human tasks (ht) to the crowd. A task's description comprises functional properties (FPs), e.g.,



(a) BPEL/B4P processes. (b) FP matching. (c) NFP matching. (d) Select and assign.

Figure 1 Enhanced B4P environment: matching, ranking, and selection of human workers.

assignment regions R, and furthermore, non-functional properties (NFPs) including capabilities C and quality expectations Q. In Figure 1b a set of potential crowd workers W that can participate in the task ht is estimated by matching the task's aligned set of regions to the regions available in the environment. Next, in Figure 1c the initial set of workers is reduced to a set that provides the required NFPs, e.g., their capabilities. Additionally, in this step the workers are ranked according to their capabilities' related skill level. The skill level hints the expectable quality. It is important to note, that the requester has no knowledge about the hierarchical structure in the crowd. Hence, the workers recommended to the requester are actually a set of coordinators with aggregated capabilities. Finally, to guarantee the promised quality in Figure 1d the human tasks are assigned according to the ranking.

All interactions within the environment are monitored by a logging component. These logs contain information regarding task creation time, task assignments, interactions among workers and so forth. Based on this information various metrics are automatically calculated such as task processing speed and reliability (accepted and finished tasks). The ranking procedure is based on these metrics which are frequently updated. Thus, this feedback-loop approach enables the system to self-adapt based on the workers' behavior.

4 Non-functional properties in B4P

Current B4P compositions include mainly functional properties. In the common case these comprise the WSDL based information (operations and ports), and, related to the potential assignments, role-based access models to the activities around the task (c.f., [3, 30, 51]). However, the dynamic nature of crowdsourcing environments requires a flexible definition of interactions in B4P. In particular, a situation aware selection of potential workers must be possible. Thus, instead of defining strict interaction models it is necessary to include in the definitions some properties that guarantee a certain degree of freedom at composition and execution time. We call these particular properties *non-functional properties* for B4P. Just as the functional properties, these properties define possible task assignments and human task processors that come into consideration. However, non-functional properties' values are either not completely known a-priori, or tend to change rather frequently over time. A crowd worker's observed performance might be better or less than expected because of the worker's current situation and context which influences her/his behavior. A typical example includes her/his present task load. The following section gives a brief overview of the concepts included in BPEL, B4P, and WS-HT before the new extensions are discussed, presented and explained in an XML sample.

4.1 Human tasks in B4P

Figure 2 shows the relation between BPEL, B4P, and WS-HT. The figure is a simplified version of the relation and contains only the essential elements that are addressed by our extensions.

Generally, a BPEL *Process* uses services to process the included activities. In our case these can either include SBSs or HPSs. In the event of an HPS activity, the B4P specification allows to bridge a BPEL *Extension Activity* to a B4P *People*

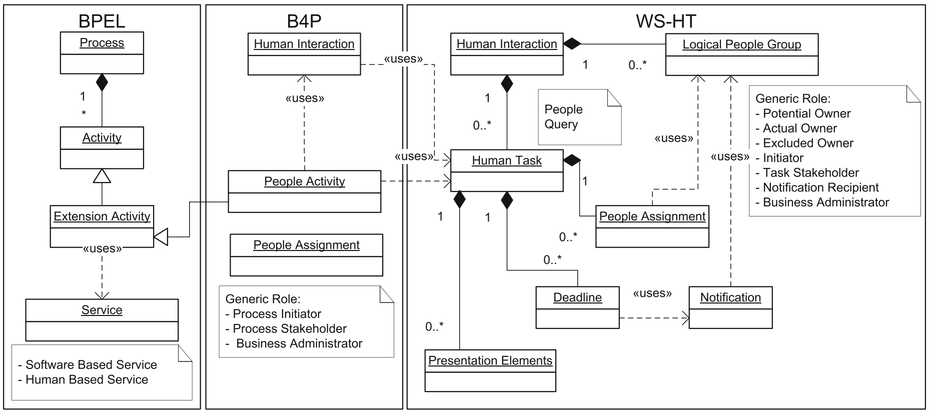


Figure 2 Simplified B4P task model.

Activity. The *People Activity* contains either locally defined tasks from B4P *Human Interaction*, a container for task definitions, or, from a WS-HT document's *Human Task*. Apart from the wrapping tasks B4P also defines *People Assignments*. These are defined role types that refer to the whole process context. Similar to B4P's definition, WS-HT has a *Human Interaction* element. It is also a container for a task collection, however, with additional elements. In particular, it allows to define *Logical People Group*s that list the involved parties and their roles aligned with the tasks. The *Human Task* used by both B4P and WS-HT is actually a WS-HT element that defines the individual assignments between task and human (*People Assignment*), deadlines with the *Deadline* element, and human readable description of the task with the *Presentation Elements*.

4.2 Basic model and extensions

The current specification of WS-HT provides no elements to include non-functional properties into their definition as required in crowdsourcing. This is a major shortcoming in the specification when applied to dynamic environments. The reminder of this section explains the extensions to WS-HT necessary to include non-functional properties. Additionally, in reference to the previous work in [37] with a similar scenario, we consider that the agreements between the crowd brokers, the previously introduced *Coordinator*s, and the customers settle on an XPath processable WSLA document. Hence, the values to the WS-HT functional and non-functional properties are manly taken from an WSLA XML document.

Human interaction Listing 1 presents an extract of a WS-HT XML document. Such a document starts with the *humanInteractions* tag and for our purpose links an additional namespace (*cp*). Next, the namespace (*tns*) related to the crowdsourcing service is defined. Then, the *import* tag specifies the WSDL file and its location for all WS operations specified in the interaction. The following *logicalPeopleGroups* and *tasks* will be detailed by the following listings. At the end, an example of a notification is listed. In WS-HT notifications are used to notify a person or a group of people of a noteworthy business event. For this crowdsourcing scenario a general one is defined

```
<?xml version="1.0" encoding="UTF-8"?>
<htd:humanInteractions xmlns:htd="http://www.example.org/WS-HT"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.infosys.tuwien.ac.at/hps/" targetNamespace="http://www.infosys.tuwien.ac.at/hps/"
  xmlns:cp="http://www.infosys.tuwien.ac.at/crowd/params" targetNamespace="http://www.infosys.tuwien.ac.at/crowd/params"
  xsi:schemaLocation="http://www.infosys.tuwien.ac.at/WS-HT ws-humantask.xsd">
  <htd:import importType="http://schemas.xmlsoap.org/wsdl/"
   location="CrowdSourcing.wsdl" namespace="http://www.infosys.tuwien.ac.at/crowd" />
  <!-- htd:logicalPeopleGroups -->
  <!-- htd:tasks -->
  <htd:notifications>
   <htd:notification name="taskComplete">
    <htd:interface portType="tns:CustomerPT"
    operation="reportComplete" />
    <htd:peopleAssignments>
     <htd:recipients>
      <htd:from>htd:getInput("SLA")//wsla:Parties/wsla:ServiceConsumer</htd:from>
     </htd:recipients>
    </htd:peopleAssignments>
    <htd:presentationElements/>
   </htd:notification>
  </htd:notifications>
</htd:humanInteractions>
```

Listing 1 Human interactions including roles, tasks and notifications.

```
<htd:logicalPeopleGroups>
  <htd:logicalPeopleGroup name="taskCoordiantors">
   <htd:documentation xml:lang="en-US">
    coordinate tasks in the crowd
   </htd:documentation>
   <htd:parameter name="region" type="xsd:string" />
   <htd:parameter name="communities" type="cp:ListOfCommunities" />
   <htd:parameter name="capabilities" type="cp:tListOfAggregateCapabilities" />
   <htd:parameter name="constraints" type="cp:tListOfConstraints" />
  </htd:logicalPeopleGroup>
  <htd:logicalPeopleGroup name="taskSupervisors">
   <htd:documentation xml:lang="en-US">
    supervises tasks in the crowd and aggregates results
   </htd:documentation>
   <htd:parameter name="region" type="xsd:string" />
   <htd:parameter name="communityId" type="xsd:string" />
   <htd:parameter name="capabilities" type="cp:tListOfAggregateCapabilities" />
   <htd:parameter name="constraints" type="cp:tListOfConstraints" />
  </htd:logicalPeopleGroup>
  <htd:logicalPeopleGroup name="taskWorkers">
   <htd:documentation xml:lang="en-US">
    can processes tasks
   </htd:documentation>
   <htd:parameter name="region" type="xsd:string" />
   <htd:parameter name="communityId" type="xsd:string" />
   <htd:parameter name="capabilities" type="cp:tListOfCapabilities" />
   <htd:parameter name="constraints" type="cp:tListOfConstraints" />
  </htd:logicalPeopleGroup>
</htd:logicalPeopleGroups>
```

Listing 2 Groups defined in the context of the crowd model.

at the end of the interaction definition. Via the port *CustomerPT* the customer is informed that her/his outsourced task is complete. In this case, the *from* tag does not specify one of the logical people groups defined in Listing 2. Instead, it specifies the service customer which can be found in the WSLA's *Parties* section.

Logical people groups This part of the human interactions definition organizes the members of the crowd in our scenario in groups of people. As motivated in Section 3.2 and in detail explained later in Section 5 we have identified three distinct roles for structured crowdsourcing. Listing 2 defines those roles for the WS-HT document. Furthermore, the content of the tag *logicalPeopleGroup* allows to characterize the different roles in the outsourcing process. Here, our first extension to the standard is evident. Apart from the standard WS-HT parameter `region`, the parameters define some necessary non-functional properties for the groups.

These include a parameter for the affiliation of the crowd members. For the coordinator this is a list of `communities`. The `communityId` is for members affiliated with exactly one community (supervisor and worker). Next, the capabilities of a member are stated in `capabilities`. Those of the coordinator and supervisor are aggregated across the hierarchy. Finally, a property denoted `constraints` allows the customer and owner of the WS-HT to state some strict or relaxed filter options on the different roles. Hard constraints may be used to hide sensitive information from crowd members and soft constraints may be used to influence the automatic matching of tasks to crowd members.

Tasks The main part of a WS-HT states the tasks and subtasks. Listing 3 includes a single document review task, a service offered by many current crowdsourcing platforms. After a brief documentation of the content, the WS port offering the review service is defined. The port connects the caller to the selected crowd coordinator. Any further delegation of the task is not directly influenced by this task specification, but remains in the hands of the crowd members. Another necessary extension to the standard is the *quality* tag. It defines the expected quality of the promised reviews. The value of the quality is derived from the publicly accessible WSLA document with immutable content and prepared to be reused for future similar review tasks. Therefore, there is a need for a document that contains the customers requirements for this particular task. In the presented case, this is the `DocumentReview` document only accessible by the customer. In the tag *people-Assignments*, after region and capabilities criteria, `constraints` tag contains a number of hard or soft constrains in *listOfConstraints* that define the final selection and are provided by `DocumentReview`. These might include constraints that must not be propagated to the crowd. Otherwise, as by definition the content of the tag *peopleAssignments* maps between the roles defined in WS-HT and their properties defined in Listing 2.

The *presentationElements* tag contains standard information about human tasks and notifications. This is another example, where all values are gathered from the customers own document. As intended by the standard, the content is human-readable information about the task and structured according to content displayable at a user interface. This allows the customer to specify the particularities of the task and the involved crowd members to deal with their tasks and notifications via a user interface.

From Listing 3 it becomes apparent that the document comprises a number of chapters. Also, urls are defined indicating where the document is provided for review (`document_url`), the location of the questionnaire (`review_url`), and a result submission portal at `result_url`. According to the WS-HT standard, the *task* tag can also include compositions with a set of subtasks connected to individual task definitions. These tags provide a convenient method to detail segmented tasks a-priori. Nevertheless, crowdsourcing is a dynamic environment and a top-down segmentation of the task might contradict the current possible worker assignments. Thus, for our scenario we transfer the burden of on-line segmentation to the involved

```
<htd:tasks>
  <htd:task name="DocumentReview">
    <htd:documentation xml:lang="en-US">
      This task requires the review of a multipage document
    </htd:documentation>
    <htd:interface portType="tns:ReviewHandlerPT"
      operation="review" responsePortType="tns:ReviewHandlerCallbackPT" responseOperation="reviewResponse" />
      <cp:quality>
        htd:getInput("SLA")//wsla:Expression[wsla:SLAParameter='quality']/wsla:value
      </cp:quality>
    <htd:peopleAssignments>
      <htd:taskStakeholder>
        <htd:from logicalPeopleGroup="taskCoordiantors">
          <htd:argument name="region"> <!-- hard constraints -->
            htd:getInput("SLA")//wsla:Expression[wsla:SLAParameter='region']/wsla:value
          </htd:argument>
          <htd:argument name="capabilities"> <!-- hard constraints -->
            htd:getInput("SLA")//wsla:Expression[wsla:SLAParameter='capabilities']/wsla:value
          </htd:argument>
          <htd:argument name="constraints"> <!-- hidden hard/soft constraints -->
            htd:getInput("DocumentReview")/listOfConstraints
          </htd:argument>
          </htd:from>
      </htd:taskStakeholder>
    </htd:peopleAssignments>
    <htd:presentationElements>
      <htd:name xml:lang="en-US">Document Review</htd:name>
       <htd:presentationParameters>
        <htd:presentationParameter name="title" type="xsd:string">
          htd:getInput("DocumentReview")/title
         </htd:presentationParameter>
         <htd:presentationParameter name="chapters" type="xsd:integer">
          htd:getInput("DocumentReview")/chapters
        </htd:presentationParameter>
         <htd:presentationParameter name="document_url" type="xsd:string">
          htd:getInput("DocumentReview")/document url
        </htd:presentationParameter>
         <htd:presentationParameter name="review_url" type="xsd:string">
          htd:getInput("DocumentReview")/review url
        </htd:presentationParameter>
         <htd:presentationParameter name="result_url" type="xsd:string">
          htd:getInput("DocumentReview")/result url
        </htd:presentationParameter>
        <!--  and more -->
      </htd:presentationParameters>
      <htd:subject xml:lang="en-US">
        Review of a document {$title} comprising {$chapters}
      </htd:subject>
      <htd:description xml:lang="en-US" contentType="text/plain">
        Review the attached document {$title} comprising {$chapters}.
        Find the document at {$document url} and the related questionnaire at {$review url}.
        Only fully and in−time completed questionnaires accepted at {$result url}.
      </htd:description>
    </htd:presentationElements>
    <!-- htd:deadlines -->
    <!-- crowdsourcing is flexible thus NO task compositions -->
  </htd:task>
</htd:tasks>
```

Listing 3 The set of human tasks.

crowd supervisors and hint a possible segmentation in this example by providing the number of chapters.

Deadlines The WS-HT standard also provides sections to notify the associated parties in the process. The section related to a particular task is enclosed in the *deadlines* tag displayed in Listing 4. Related to our document review example, the defined notification chain is triggered by a completion deadline. The deadline itself has been agreed in the WSLA. An escalation is triggered if the condition stated is violated. In the example `DocumentReview`'s value `reviewComplete` is set to true if the complete review has been submitted to the aforementioned `result_url`. If the condition is broken then the assigned supervisors and coordinators are informed about the SLA violation.

```
<htd:deadlines>
  <htd:completionDeadline name="notifyManagement">
    <htd:documentation xml:lang="en-US">
      notify the requester on deadline
        </htd:documentation>
    <htd:for>htd:getInput("SLA")//wsla:Expression[wsla:SLAParameter='deadline']/wsla:value</htd:for>
    <htd:escalation name="deadlineMissReview">
        <htd:condition>
          <![CDATA[ htd:getInput("DocumentReview")/reviewComplete = true() ]]>
        </htd:condition>
      <htd:notification name="deadlineMissSupervisor">
        <htd:documentation xml:lang="en-US">
          inform the hierarchy of responsible roles
            </htd:documentation>
        <htd:interface portType="tns:ReviewHandlerPT" operation="escalate" />
        <htd:peopleAssignments>
          <htd:recipients>
            <htd:from logicalPeopleGroup="taskSupervisors">
              <htd:argument name="supervisorID">
                htd:getActualOwner("AssignedSupervisor")
              </htd:argument>
            </htd:from>
            <htd:from logicalPeopleGroup="taskCoordiantors">
              <htd:argument name="coordinatorID">
                htd:getActualOwner("AssignedCoordinator")
              </htd:argument>
            </htd:from>
          </htd:recipients>
        </htd:peopleAssignments>
        <htd:presentationElements/>
      </htd:notification>
    </htd:escalation>
  </htd:completionDeadline>
</htd:deadlines>
```

Listing 4 Defined timeouts and escalation actions.

5 Social aggregator

Today, social networks are a mass phenomenon found in private (e.g., Facebook) and professional environments (e.g., LinkedIn). It is reasonable to assume that the trend of social networks will continue to penetrate more and more aspects of our lives. In a business context a social network is either formed explicitly, by manually adding contacts, or implicitly, based on observed interaction and collaboration patterns. We argue that it is highly beneficial to consider social aspects in crowdsourcing since a clique in a social network is more likely to efficiently work on collaborative tasks than a group of random workers. On a global scale, members of the latter are likely to have never worked together before, to have different cultural background, to speak a different language, and to live in different timezones, which altogether makes it extremely hard to create high-quality task processing results.

The structure of companies is typically organized hierarchically. We borrow that concept to some extent and distinguish between three roles in our crowdsourcing system:

– *Workers* perform the actual processing of tasks and are assigned to one or multiple supervisors.

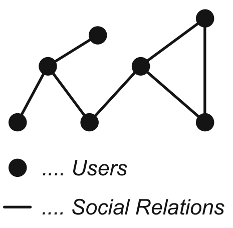
- *Supervisors* represent the head of a group of workers. They are responsible for breaking a task down into subtasks, to distribute those subtasks to suitable workers in her/his team, and to finally check the result. Each supervisor in turn is assigned to one or multiple coordinators.
- *Coordinators* are the interface between the customers who submit tasks to the social network and the supervisors who are responsible for tasks processing.

Figure 3 shows the main steps that are performed to augment the social network in order to make it suitable for crowdsourcing. The origin is a social network (c.f., Figure 3a) that was formed either explicitly or implicitly; the nodes denote users of the crowdsourcing system and the edges social relationships between those users. Every user has a profile describing her/his skills. In the next step, illustrated in Figure 3b, role hierarchies are formed.

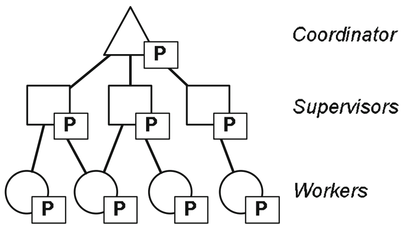
We use *betweenness centrality*, a measure from graph theory indicating the importance of a node, to determine a member's role. We propose betweenness centrality because it is often used in social and communication networks to estimate the potential monitoring and control capabilities a node may have on data flowing through the network [15]. In particular, we assume that nodes obtaining the role of a supervisor will have a high betweenness centrality value because these nodes have great influence on task flows. Let us define the graph $G(V, E)$ consisting of the set of vertices V and the set of edges E. Shortest-path betweenness centrality, as used in this work, defines the importance of a node s based on how many pairs of vertices go through $s \in V$ in order to connect through shortest paths in G (e.g., see [10, 47]). Betweenness centrality $B(s)$ is formally defined as:

$$B(s) \leftarrow \sum_{u \neq s \neq t} \frac{g_{ut}(s)}{g_{ut}} \tag{1}$$

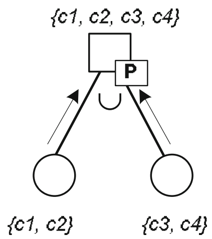
where g_{ut} is the number of shortest paths linking nodes u and t; and $g_{ut}(s)$ is subset of those paths that contain node s. When the betweenness centrality of a user exceeds a certain threshold τ_S s/he has the prerequisites for becoming a supervisor, if it is greater than an even higher threshold value τ_C s/he could adopt the role of coordinator. This functionality is outlined in Algorithm 1.



(a) Social network.



(b) Role hierarchies.



(c) Aggregators.

Figure 3 Social roles and aggregation.

When the importance values are calculated and a user fulfills the basic requirements for becoming a supervisor or coordinator two different approaches are supported how to actually decide on the roles:

- *Invitation:* The platform invites users exceeding a certain betweenness centrality threshold to adopt a hierarchically higher role. The user may accept or decline the offer.
- *Nomination:* The platform only nominates candidates for higher roles based on their importance in the social network. Users connected to the nominee can vote whether they support the candidate. Only with a certain minimum number of supporters the user is awarded the higher role. This prevents to assign high roles to users who have a high number of relationships and therefore a high importance indicator but whose relationships are mostly superficial and weak.

The final step to make the social network 'crowdsourcing-ready' is to create the higher-role profiles as an aggregation of all affiliated user profiles, as seen in Figure 3c. This provides the basis for simple and rapid matching of tasks to competent groups in social crowdsourcing, as described in the following section.

Algorithm 1: Detecting roles of users.

```
input  : The social collaboration graph $G(V, E)$.
output: The set $V$ containing workers $u \in V$ with different roles in the social network.
1  $\tau_S$ /* threshold for supervisors */
2  $\tau_C$ /* threshold for coordinators */
3  /* simplified betweenness centrality evaluation of nodes in network $G$ (for details see [10]) */
4  for $v \in V$ do
5      foreach $n \in N(v)$ /*neighbors of v*/ do
6          if shortest path through $n$ then
7              /* save distance */
8          end
9      end
10 end
11 foreach node along shortest path to $v$ do
12     /* from the most distant $s$ to $v$ */
13     if $s \neq v$ then
14         increase $B(v)$
15     end
16 end
17 for $u \in V$ do
18     if $B(u) \geq \tau_S$ then
19         $isSupervisor(u, true)$
20     end
21     else if $B(u) \geq \tau_C$ then
22         $isCoordinator(u, true)$
23     end
24 end
```

6 Task segmentation and matching

In this section we detail our approach for task processing in the crowd. First, we explain in detail how human tasks (as defined in the context of B4P) are passed from coordinators to supervisors and finally assigned to workers. As next step, we introduce a ranking approach to select the best suited coordinator.

6.1 Hierarchical activities

Crowd activities can be structured as hierarchies (see Figure 4) using *parent* and *child* relations. Child activities specify the details with respect to the (sub-)steps in collaborations, for example, sub-activities in the scope of a parent activity. This allows for the refinement of collaboration structures as the demand for a new set of activities (e.g., performed by different people and services) increases. The need for the dynamic refinement of collaboration structures is especially required when people have limited experience (the history of performed activities) with respect to a given objective or goal [42]. Furthermore, some people tend to underestimate the scale and complexity of an activity; thus the hierarchical model enables the segmentation of activities into sub-activities, which can be, for example, delegated to other people.

Activities have a *relatedTo* property which provides a mechanism to link to any other activity. Typically, multiple members work on the same activity with different roles. The *InvolvementRole* identifies the *coordinator*, *supervisor*, and responsible *worker* of an activity. Involved workers apply a set of *GenericResources* to perform their work. Objects such as documents are represented as a shared *Artifact*. A *ControlAction* captures activity-change events, interactions between members, and work carried out. Actions can trigger events describing the progress of activities.

6.2 Social interactions

Figure 5 and related sub-figures show how task processing is performed in the crowd. We argue that interactions in crowdsourcing environments are governed by *user preferences* and *social trust*.

Crowdsourcing follows the open world assumption which permits users to join and leave a particular community (platform) at any time. Instead of following a set of company rules or policies, crowd workers can be regarded as 'self-employed' individuals. However, it is in the interest of the worker to earn higher rewards and to work on tasks matching her/his expertise and interest. Also, we believe that complex tasks are typically rewarded higher as compared to simple tasks. Crowd members may decide to work with other members on a joint task based on previous experience or recommendations received from friends. Indeed, these interactions are not known in advance. Therefore, it is not possible to specify different task processing patterns

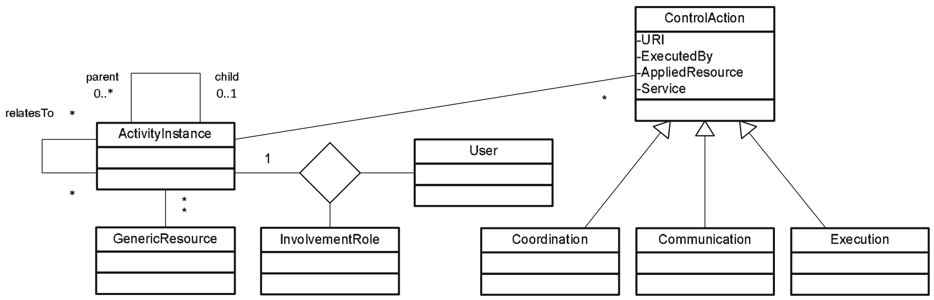
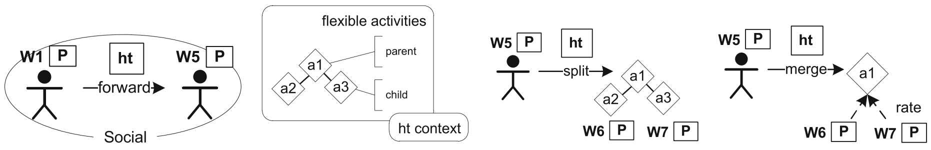


Figure 4 Excerpt of hierarchical activity model.



(a) Forward. (b) Crowd activities. (c) Assign. (d) Merge and rate.

Figure 5 Segmentation of tasks in social network.

that are performed in the crowd at the B4P or WS-HT level. The segmentation of human tasks is illustrated by Figure 5.

As a first step in Figure 5a we assume that the selected coordinator W1 forwards (e.g., through delegation) the human task to W5 who is a supervisor. The selection of the supervisor may entirely depend on W1's preferences to forward tasks to W5. Moreover, the previously discussed preferences as defined along with the groups may prevent W1 to forward a task to any of the supervisors s/he is connected to in the social network.

The supervisor receives a given task and performs some segmentation. In this context, we introduce *crowd activities*, which are collaborative activities performed by crowd members in a flexible manner. The notion of flexible (crowd) activities is independent of the previously discussed process activities that are designed in the context of processes such as BPEL. Here, we show how to combine *flexible* interactions and *top-down* process activities and tasks in order to support adaptive compositions of human- and software-based services. In our previous work [42] we have designed an activity model supporting collaborative working environments. The model and a set of collaboration tools have been implemented on top of Web services technology. Also, the inclusion of human capabilities in service-oriented collaboration environments is supported through the HPS concept [45]. These activities include, for example, 'write document', 'review document', 'proof-read paragraph'. Furthermore, these activities can be created and modified on-demand by people, e.g., the supervisor, based on their preferences and expertise in performing a specific type of task. Figure 5b shows an example of such an activity structure that can be created in a specific human task (ht) context.

Assume the 'review document' task (cf. Section 4) that needs to be outsourced to the crowd. The supervisor decides to split and to process the task by creating a hierarchical crowd-activity structure. A *parent* activity is initiated with the task's context data (presentation, elements, time constraints, etc.). Depending on the task's properties (e.g., duration, effort) sub-activity a2 and a3 are associated as *child* activities to a1. The segmentation step may be assisted by an *activity service* (a software service to manage crowd-activities) that recommends how many sub-activities the parent activity should be segmented into. Though, it is the responsibility of the supervisor W5 to allocate sub-activities to workers.

In social environments selection preferences and resulting interactions typically depend on the *trust* between actors. How much W5 trusts its neighboring peers (e.g., workers) is strongly influenced by previous interaction behavior. For example, W5 may trust a worker W6 more than W7 in performing a given activity depending on W5's collaboration experience. Positive experience results in higher trust between collaboration partners. We have established a set of metrics to measure collaboration experience (see [41, 43, 49]) including the *activity success* and *responsiveness* when processing an activity. A detailed description of these metrics and a trust model is, however, out of scope of this work. The assignment procedure is shown in Figure 5c where W5 assigns a2 to W6 and a3 to W7. Each of these sub-activities can be controlled (e.g., inspecting the status and progress of an activity) by the supervisor. Once the workers W6 and W7 (see Figure 5d) deliver the results, the supervisor takes the output of a2 and a3 and merges them. For example, the results can be combined by simply merging separate document sections to one document that was reviewed by W6 and W7. However, since the supervisor W5 is responsible for the final quality, W5 checks the result before the output of a1 is returned to the coordinator and/or B4P requester. How the result is being passed from the supervisor to the B4P requester may in fact depend on the 'social protocol' or preferences of actors. A coordinator may prefer to act as the main interface towards the crowd and thus may want to return the result.

The final step is the rating of the supervisor. Rating is performed to give feedback how well the supervisor distributes activities in the crowd. Crowd workers will be satisfied if the supervisor distributes activities that fit their expertise. Also, a worker's queue should neither be empty nor overloaded. This means that the supervisor should not accept too many tasks to avoid overload conditions. Careless assignments (e.g., activities that have low or no overlap with a worker's interest and skills) and false assumptions with regards to activity effort would cause bad ratings.

Next, we will introduce a ranking algorithm to rank coordinators based on capabilities and quality constraints specified by the B4P requester.

6.3 Ranking coordinators

Here we introduce our novel ranking approach that bases its input on *skill information* as well as *social network metrics*. The approach consists of three essential steps that are briefly introduced in Algorithm 2.

Algorithm 2: Ranking approach outline.

```
input  : The social graph $G(V, E)$ and detected roles.
output: Ranked list of coordinators.

1. Calculate importance scores in the hierarchical social network (detailed by Algorithm 3).
2. Calculate the rank of supervisors $SR$ based on
     – their skills and
     – their social standing (reputation) within the social network.
   Also, calculate the rank of each worker based on
     – their skills and
     – task load conditions.
   Append the workers' rank connected to a particular supervisor to $SR$.
3. Each coordinator gets the ranking scores of the top-ranked supervisor it is connected to. Sort coordinators
   according to their ranking score.
```

Let us start with the definition of procedure to rank the importance of individuals in social networks. In this work we utilize the concept of hubs and authorities in Web-based environments. This concept has been introduced by Kleinberg [23] to rank Web pages in search queries using the HITS algorithm. The notion of authorities in social or collaborative networks can be interpreted as a measure to estimate the relative standing or importance of individuals in social networks. Compared to methods such as PageRank [34], the main advantage of the HITS model is that hub and authority scores are calculated for each node in the network.

Applying this idea in our scenario, assume an undirected social network and roles of users that were detected using the previously introduced approach (cf. Algorithm 1). Also assume the hierarchical network that can be created considering the roles and the social graph. Coordinators are responsible for forwarding tasks to supervisors, thereby acting as *hubs* in the network. According to the theory developed in [23], the hub $H(u)$ and authority $A(u)$ scores of nodes $u \in V$ in the network is calculated as:

$$H(u) \leftarrow \sum_{(u,v)\in E} A(v) \qquad A(u) \leftarrow \sum_{(v,u)\in E} H(v) \tag{2}$$

However, we assume multiple roles in the social network such as coordinators, supervisors, and workers. Workers develop expertise in different areas depending on their interest and task processing behavior. Given the model in (2), workers would have higher authority scores if they receive requests to perform activities from supervisors that are connected to many 'good' authorities (i.e., workers). Good workers are characterized by their reliable task processing behavior which is monitored at runtime.[10]

Thus, a supervisor trusts a particular worker if the worker processes tasks in timely and satisfactory manner. Supervisors are rated by workers based on the suitability of assigned work. For example, supervisors who carelessly delegate tasks to workers without knowing their interests or who ignore the workers' load conditions (e.g., oversupply of task assignments in too short time frames) would receive bad ratings. Ratings thereby influence the weight of a relationship between worker and supervisor. Notice, both trust and rating relations are established upon interaction monitoring and mining. Thus, in addition to the undirected social links, directed relations are introduced based on collaborations between actors in the crowdsourcing environment.

The recursive definition of hub and authorities is typically computed using an iterative algorithm. In the following Algorithm 3 we introduce an extended HITS algorithm suitable for calculating hub- and authority scores in hierarchical social networks.

The goal of the algorithm is to calculate:

- hub scores for coordinators as they forward tasks to supervisors through delegation actions,
- hub scores for supervisors as they perform pre-processing of tasks and create flexible crowd-activities that are distributed and assigned to workers,
- authority scores for supervisors as they receive task requests from coordinators, and
- authority scores for workers as they perform the actual work.

[10]Task processing behavior is observed through interaction monitoring techniques. Interaction metrics are established to obtain weighted social relations between actors. These relations are used to automatically calculate social network metrics such as hub- and authority-based importance scores.

The first step in Algorithm 3 (see lines 2–4) is to initialize two vectors $\mathcal{H}$ and $\mathcal{A}$ that hold hub and authority scores, which are updated after each iteration t. Without any prior (node bias), the initialization vectors p^H and p^A hold for each node the same initial value. The main loop (lines 6–26) is executed until the ranking scores converge (i.e., the ranking order of nodes is no longer changed between the step $t-1$ and t). For each node $u \in V$, we update hub- and authority scores according to the aforementioned update procedure. The set of u's neighbors is obtained by $N(u)$. Assume that u is a coordinator (line 10), then only u's hub score $\mathcal{H}(u)$ is updated. The next case holds (line 13) if u is a supervisor and the neighbor v is a worker. In this case, the supervisor's hub score needs to be updated.

Algorithm 3: HITS algorithm in hierarchical social networks.

```
1  /* Initialize hub and authority scores */
2  for $u \in V$ do
3      $\mathcal{H}(u)^{(0)} \leftarrow p_u^H, \mathcal{A}(u)^{(0)} \leftarrow p_u^A$
4  end
5  $t = 1$
6  while not converged do
7      for $u \in V$ do
8          /* update ranking scores */
9          for $v \in N(u)$ do
10             if $isCoordinator(u)$ then
11                 $\mathcal{H}(u)^{(t)} \leftarrow \mathcal{H}(u)^{(t-1)} + \mathcal{A}(v)^{(t-1)}$
12             end
13             else if $isSupervisor(u) \wedge \neg isCoordinator(v)$ then
14                 $\mathcal{H}(u)^{(t)} \leftarrow \mathcal{H}(u)^{(t-1)} + w_{vu}\mathcal{A}(v)^{(t-1)}$
15             end
16             else if $isSupervisor(u) \wedge isCoordinator(v)$ then
17                 $\mathcal{A}(u)^{(t)} \leftarrow \mathcal{A}(u)^{(t-1)} + \mathcal{H}(v)^{(t-1)}$
18             end
19             else if $isSupervisor(v)$ then
20                 $\mathcal{A}(u)^{(t)} \leftarrow \mathcal{A}(u)^{(t-1)} + w_{vu}\mathcal{H}(v)^{(t-1)}$
21             end
22         end
23     end
24     /* Normalize rankings and test for convergence */
25     $t = t + 1$
26 end
```

As mentioned before, the hub score of supervisors is influenced by ratings they receive from workers. Thus, the authority score of v is added with weight w_{vu} to $\mathcal{H}(u)$ (see line 14). In case v holds the role of a coordinator (line 16), u's authority score is updated. Notice, weights are only calculated between supervisors and workers as we assume stronger collaborations between these two actors whereas coordinators mainly act as 'entry points' to the crowdsourcing platform.

The final procedure (line 19–21) is performed to update the worker's authority score. Here, the score $\mathcal{H}(v)$ is appended with weight w_{vu} that is calculated based on mining metrics (e.g., how much the supervisor v trusts the worker u). After these steps, a check in line 24 verifies if convergence has been reached. For larger social networks a fixed number of steps can be used to reduce the time needed for computing importance scores. After convergence the final scores are copied into H and A.

Next, we introduce the ranking procedure to process crowd-activities that can be segmented into flexible activity structures. Certain activities can be decomposed hierarchically into sub-activities depending on their required processing effort or complexity. Algorithm 4 shows the procedure to rank coordinators. As input we assume the social network graph $G(V, E)$, an activity $a \in A$ which could be part of a complex activity structure, and the set V^C of coordinators. The goal of the algorithm is to assign a ranking score to each user $u \in V^C$. First, a set V_u^S is initialized that contains supervisors connected to u (see lines 2–7). The next loop (lines 8–21) shows how to calculate rankings scores for supervisors. Note that the coordinator u acts as a 'proxy' for supervisors; thus u's score is based on the score of the highest ranked supervisor (lines 22–23).

Algorithm 4: Rank coordinators based on social graph G.

```
input : $G(V, E)$ representing the social network graph, splittable activity $a \in A$ and the set of coordinators $V^C$
        that have already been filtered based on hard constraints.
output: Ranked list of coordinators (CR).
1  for $u \in V^C$ do
2      /* Get supervisors connected to $u$ */
3      for $v \in N(u)$ do
4          if $isSupervisor(v)$ then
5              $V_u^S \leftarrow V_u^S \cup v$
6          end
7      end
8      for $v \in V_u^S$ do
9          /* Get workers connected to $v$ */
10         for $n \in N(v)$ do
11             if $\neg isCoordinator(n)$ then
12                 $V_v^W \leftarrow V_v^W \cup n$
13             end
14         end
15         $SR(v, a) \leftarrow \alpha \cdot skill(v, a) + (1 - \alpha) \cdot (0.5 \cdot A(v) + 0.5 \cdot H(v))$
16         for $n \in V_v^W$ do
17             $score_n \leftarrow \beta \cdot skill(n, a) + (1 - \beta) \cdot (1 - getRate(n, a, DueAt(a))$
18             $score_{V_v^W} \leftarrow score_{V_v^W} + \frac{1}{|V_v^W|} score_n$
19         end
20         $SR(v, a) \leftarrow SR(v, a) + score_{V_v^W}$
21     end
22     $s \leftarrow getTopRanked(SR(a))$
23     $CR(u, a) \leftarrow getScore(s, a)$
24 end
25 /* Order by soft constraints: sort CR in descending order */
26 return CR
```

The basic idea to calculate rankings for supervisors has been shortly explained in Algorithm 2. Our approach (see Algorithm 4) is to rank supervisors based on the actual (observed) skills (line 15), the importance of supervisors in the social networks (line 15), and the actual skills of workers a supervisor is connected to (lines 16–19). The set of workers V_v^W connected to v is first initialized (lines 9–14). In the following (lines 15–20) the computation of v's ranking score is shown. The initial supervisor ranking score SR is calculated as shown in line 15. The initial score is the aggregation (weighted by the parameter α) of the supervisor's skills and (social) importance. These input parameters are detailed in the following Table 1.

Table 1 Description of calculations and equations.

Ranking input	Description
Skill	The skill of supervisors and workers. The function *skill*(v, a) returns v's skill level with respect to an activity a. For example, if a demands for language skills in English with the level 'expert' and v's experience in English is at the level 'expert', v's skills match perfectly a's skill requirements. Also, skill profiles are automatically maintained by updating the users' experience. In our previous work we have designed and implemented algorithms for profile matching [44] and skill updates [40]. Details regarding skill matching and update are not presented in this work.
Importance	The relative standing of a user within the social network. As explained before, the importance of a node is based on the concept of hubs and authorities in social networks. The supervisor's importance is determined by both hub and authority scores due to the hierarchical nature of the previously explained social (collaborative) network. Hence, v's importance score is the weighted sum of the authority score $A(v)$ and the hub score $H(v)$. Different weights could be used to assign preferences to either score.

The next steps (lines 16–19) in Algorithm 4 is to calculate ranking scores for each worker connected to the supervisor v. This is done in a similar manner as for the supervisor. However, instead of considering the importance of a worker within the social network, we take other workload related factors into account. The function *getRate* calculates the workers' rate based on the earliest possible start time (influence by the workers' task queue size) and activity effort. This means that even if a supervisor has high skills and high importance, it still needs to be connected to a set of workers who have free resources in terms of free time to process a crowd activity. Otherwise, the supervisor would need to handle all activities him/herself. The score of each worker is appended with equal weight $\frac{1}{|V_v^W|}$. The final score SR is the sum of the supervisor's inital ranking score plus the workers' ranking scores.

7 Implementation and evaluation

Our evaluation and results are based on a proof of concept implementation of various introduced concepts and simulations of interactions in social-crowd environments. The following Section 7.1 describes the SOA-based crowdsourcing environment including the lifecylce of a human task and the principle interactions between services, Section 7.2 explains how the basic social network structure has been generated and Section 7.3 presents our findings.

7.1 SOA-based crowdsourcing environment

In this section we provide an overview of the main services and the most important interactions between services (see Figure 6). The implementation of our NFP-aware B4P execution environment is mainly built on-top of a service-oriented collaboration environment. The collaboration services, however, can be used independently of any top-down process model. The main extensions of the environment consist of the `WS-HT Interface` (a plugin of the `HPS Middleware`) to provide a bridge

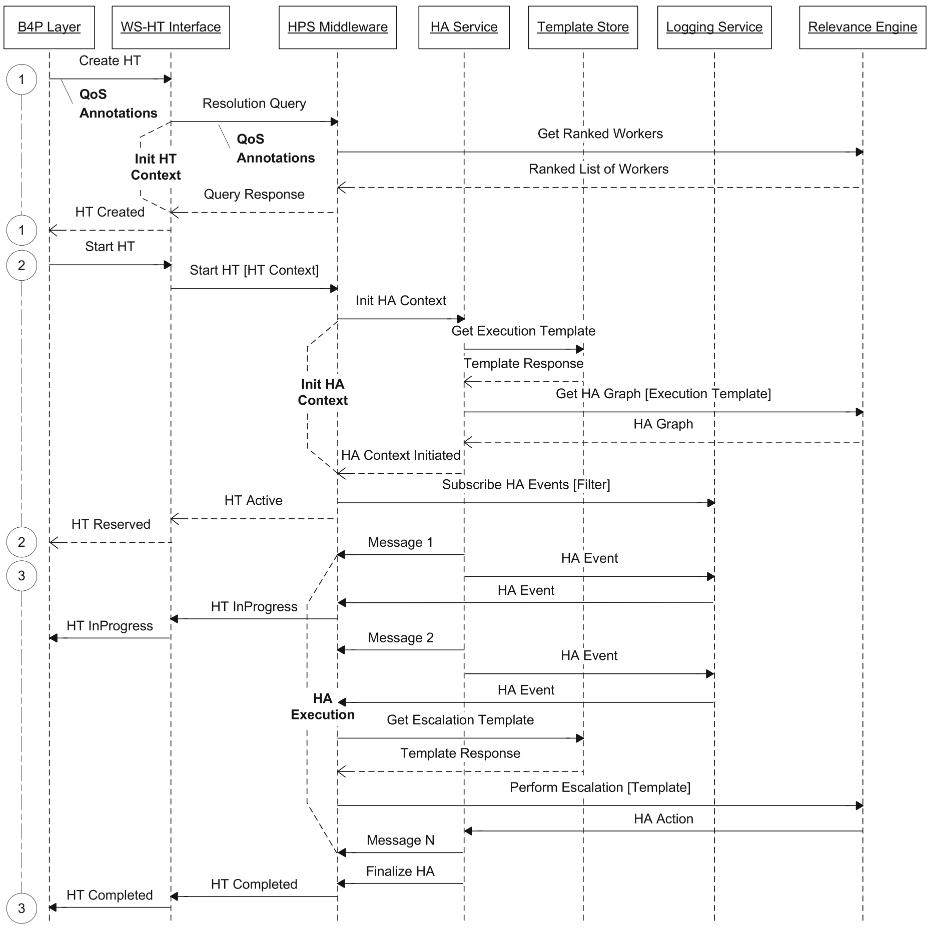


Figure 6 Sequence diagram of adaptive B4P process execution.

between B4P and the crowdsourcing environment. The protocol between the `B4P Layer` and the `WS-HT Interface` is in conformance with the WS-HT [4] standard.

The collaboration environment consists of a SOA-runtime for mixed service-oriented systems (see `HPS Middleware`). Unlike traditional SOA-based systems, also human-based services (i.e., HPSs) are made available for discovery and invocation [45]. Coordination and collaboration among people and services (HPS and SBS) is achieved by using an activity service (`HA Service`). The `Template Store` contains activity skeletons (e.g., activity structure) that can be instantiated at runtime. Such templates include, for example, the definition of parent child activities to perform a document review. The `Logging Service` monitors all interactions and saves XML-messages and additional metadata in a database for later analysis. The `Relevance Engine` implements ranking and mining algorithms.

The lifecycle of human task execution is structured into three essential phases. First, a resolution query is performed to find suitable candidate workers who can process a human task. Second, a crowd-activity structure is initialized that allows

```
<mex:Metadata>
 <mex:MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:definitions>
   <!-- Omitted -->
  </wsdl:definitions>
 </mex:MetadataSection>
 <mex:MetadataSection Dialect="http://xmlns.com/foaf/0.1/">
  <rdf:RDF xmlns:foaf = "http://..." xmlns:capability = "http://.../capability.owl#">
  <foaf:Person rdf:about="http://www.infosys.../staff/">
   <foaf:name>H. Psaier</foaf:name>
    <foaf:interest rdf:resource="http://.../hpsaier/interests.rdf"/>
    <!-- Omitted -->
    <capability:op>
     <capability:port id="TSportType">
      <capability:op id="translateDoc">
       <capability:opwsdlxpath>
        wsdl:operation/[@name="TSportType"]
       </capability:opwsdlxpath>
       <capability:opmetricgrounding
        rdf:resource="http://.../grounding-translateDoc.xml"/>
        <capability:opmetric>
         <capability:opmetricid>cost</capability:opmetricid>
         <capability:opmetricvalue>100.0</capability:opmetricvalue>
        </capability:opmetric>
       <capability:opmetric>
        <capability:opmetricid>reliability</capability:opmetricid>
        <capability:opmetricvalue>0.8</capability:opmetricvalue>
       </capability:opmetric>
        ...
      </capability:op>
     </capability:port>
  </foaf:Person>
  </rdf:RDF>
 </mex:MetadataSection>
</mex:Metadata>
```

Listing 5 HPS metadata exchange description.

crowd-members to process activities in a flexible manner. Third, workers collaborate to jointly work on activities (collaboration phase). Figure 6 details the interactions between the various services.

7.1.1 Human task creation and resolution of workers

A request to create a human task that is to be performed by the crowd is initiated by the `B4P Layer`. This layer is typically implemented as an extension of a BPEL orchestration engine. The specification of a human task contains additional elements to ensure the quality of a task's result (cf. **QoS Annotations**). These annotations have been introduced in the context of Listing 3 and define the required set of human capabilities, which are matched against capability profiles, and the required quality. NFP elements such as human capabilities are used in the matching procedure (see arrow *Resolution Query*).

Listing 5 shows the simplified structure of the resolved HPS information. NFP elements are embedded in the HPS's WSDL interface. In addition, an extended FOAF description is inserted into a *WS-Metadata-Exchange*[11] (MEX) document (see also [45]). The HPS framework uses SPARQL to define search queries[12] on FOAF structures. The sample response message to a MEX GET request in

[11]http://www.w3.org/Submission/WS-MetadataExchange/

[12]http://www.w3.org/TR/rdf-sparql-query/

Listing 5 comprises the following elements. The main response body contains the currently offered operations in a WSDL (omitted for brevity) and the related NFPs in the second `MetadataSection` in FOAF format. The elements with the capability prefix provide the current NFP values for a related operation defined in the WSDL section. In our current implementation, such NFPs are costs and primarily quality metrics, such as the HPSs reliability and responsiveness. The XPath statement identifies an operation uniquely. The following metric grounding resource `opmetricgrounding` links a document with metric definitions (meaning, measurement, unit, range of values, etc.) to the listed metric ids. The `HPS Middleware` interacts with the `Relevance Engine` to obtain a ranked list of workers.[13] The successful result of this interaction is denoted by the arrow *HT Created*.

7.1.2 Reserve human task and initialize activity structure

The activity structure is being initialized by *Start HT*. The `WS-HT Interface` passes the HT Context to the `HPS Middleware`, which in turn signals *Init HA Context* to the `HA Service`. Depending on the selected HT Context, different activity execution templates can be selected (*Get Execution Template*). An execution template may define how activities are processed. For example, if the result that is provided in the context of a specific human task has always low quality, an additional quality assurance step can be inserted dynamically in the execution template. The next step is to assign people to activities that are part of the execution template (see *Get HA Graph*). Ranking of people is performed by the `Relevance Engine`[14] (cf. to discussions related to matching and ranking in the previous section). The `Logging Service` logs all service interactions (i.e., SOAP calls) and also events triggered by the activity service. Activity events are fired based on activity changes (start, suspend, or finalize activity) and actions taken by human actors. Such actions include delegations of activities or the assignment of new activities. The `Logging Service` implements a publish/subscribe mechanisms that allows subscribers to get notified about specific events. The `HPS Middleware` subscribes to activity change events to monitor the status of activities (see arrow *Subscribe HA Events*). The result of these steps is *HT Reserved*.

7.1.3 Task execution and escalation handling

In service-oriented systems, people interact and collaborate by using tools and services to perform their work. Each service call (performed in the context of an activity) is processed by the `HPS Middleware`. The middleware implements a SOAP dispatcher that performs message inspection and routing. The `HA Service` notifies the `Logging Service` about activity changes (see *HA Event*). Here the activity status is changed to ‘activity in-progress’. The event is also sent to the middleware which signals *HT InProgress*. A series of messages 1 ...N is then exchanged between the `HA Service` and the `HPS Middleware` until an activity is finalized. Escalations are defined in the context of a human task (cf. Listing 4).

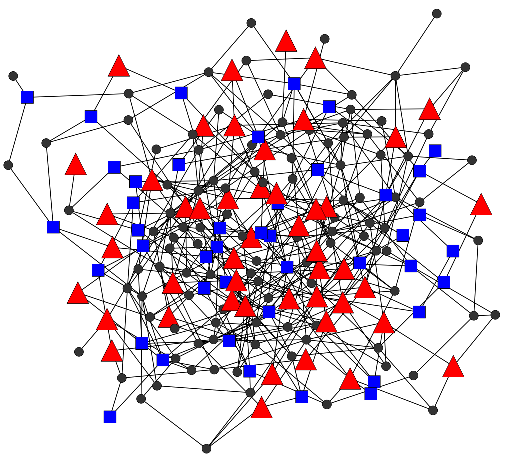
[13]For simplicity, we do not discuss the different social roles such as coordinators or supervisors in this context. Notice, the result of a resolution query is a list of coordinators if the task can be segmented in multiple crowd-activities.

[14]The `Relevance Engine` has by default access to all logs and events collected in the environment.

As mentioned before, the `HPS Middleware` acts as a bridge between the B4P-based process the activity-based collaboration services and tools that are used by crowd workers. Thus, the middleware monitors the status of activities and checks whether deviations in the progress of activities may cause deadline violations. The `Relevance Engine` receives a *Perform Escalation* call to trigger a *HA Action* if a deadline is going to be violated. As shown previously in Listing 4, a notification may be the result of such an escalation action. The `Relevance Engine` performs the escalation by sending the *HA Action* to the activity service. Notice, escalations are not directly performed by the `HPS Middleware`. The `Relevance Engine` deals with escalations to support dynamic aspects (e.g., adaptive notification chains) and also future extensions of our approach such as complex event processing features. *HT Completed* is triggered once *Finalize HA* is received from the activity service.

7.2 Social network generation

At the time when performing this research, a sufficiently large crowd user-base was not available to perform tests with real users. In our experiments, we generate synthetic social graphs to test the applicability and effectiveness of our proposed ranking model. We use two different methods to generate social graphs: *random graphs* [31, 33] are generated and graphs based on the *preferential attachment model* [7, 38]. The more general case are random graphs wherein each pair of nodes has an identical, independent probability of being joined by an edge. Preferential attachment results in more specific graphs wherein nodes preferentially connect to existing nodes with high degree (the 'rich get richer'). By using two these methods, we are able to evaluate the effectiveness of our ranking approach by considering different social network structures. Figure 7 shows a basic social network structure that has been generated according to the statistical properties as found in freely emerging networks. Each figure visualizes a graph with 200 workers.


(a)

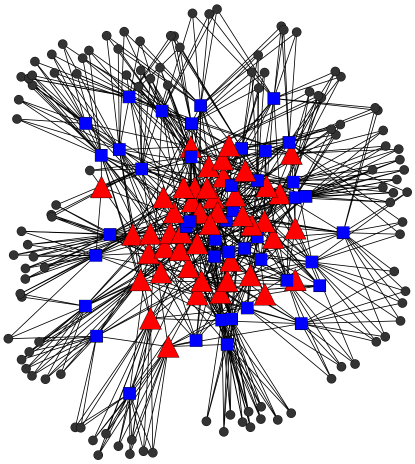

(b)

Figure 7 Generated social graphs: **a** sparsely connected random graph and **b** preferential attachment graph.

Here we employ two methods to generate social graphs:

1. *Random graphs* are based on the assumption that any random actor will establish a connection to some other random actor with probability p. The resulting graph structure is visualized by Figure 7a. In our experiments, we use a probability of 0.3 that an actor u will establish a connection with a random actor v.
2. *Preferential attachment graphs* are based on the assumption that networks emerge according to the rule of preferential attachment [38]. This process produces a scale-free graph with node degrees following a power-law distribution. The resulting social graph represents very well the structure of autonomously forming collaborations in cooperation networks [32].

By using a probability of 0.3 to generate random graphs, both graphs, random and preferential, have approximately the same amount of edges; thereby making the both types of graphs comparable with regards to number of workers and number of edges. Roles in the social network were detected according to Algorithm 1. Coordinators are visualized as triangular shapes, supervisors are depicted by rectangles, and regular workers are shown as circular nodes. One can see that the random graph in Figure 7a exhibits only sparsely connected nodes when compared to Figure 7b. Using these two graphs, we are able to compare the results of our ranking approach under different conditions. This is an important issues because sparse networks are a natural phenomenon in newly established social networks.

In each network, workers have certain skills associated with it. In our experiments we only use a single skill whose skill level is distributed according to a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with a mean value $\mu = 0.6$ and a standard deviation $\sigma^2 = 0.25$. The parameters of this model (mean value and standard deviation) yield the following skill level properties of the resulting worker population: most workers have good skills in performing their tasks with an average skill level of 0.6, some workers are highly skilled with a maximum skill level of 1.0 (top expert) and on the contrary some workers have a very low skill level (in our experiments the minimum skill level was 0.02). If a higher or lower average value would be chosen, the expected quality of returned tasks can also be expected to be higher or lower respectively. If a higher standard deviation is chosen, the likelihood of having more highly skilled workers as well as workers with very low skills increases. By choosing a lower standard deviation, it is more likely that the workers will have the average skill level of 0.6 and it is less likely that workers have high or low skills.

7.3 Discussion

We performed several experiments and compared the quality of task results considering task processing with and without social network structures. The default option of our simulation is to process activity in the context of a human task without advanced processing. This configuration provides the baseline results for comparison with the advanced processing option. The configurations of our experiments are detailed in Table 2. The entry *advanced processing* indicates whether certain activities were split and processed collaboratively in social networks.

Table 2 shows three pairs of experiments (1, 2), (3, 4), and (5, 6). Each pair compares the default processing behavior with the advanced processing option. Advanced processing means that actors' behavior is guided by their social role.

Table 2 Configurations for different experiments.

Configuration	1	2	3	4	5	6
Number of workers	100	100	100	100	200	200
Activities per round	5	5	10	10	5	5
Advanced processing	No	Yes	No	Yes	No	Yes

Coordinators forward task requests to supervisors which split tasks into multiple (crowd-)activities that are assigned to workers. In our simulation, tasks are issued by the B4P requester in fixed rounds. In each round, 5 tasks are issued in configuration 1 and 2 and also in 5 and 6. The configurations 3 and 4 are based on 10 tasks per round to analyze processing behavior (e.g., quality) under different load conditions.

7.3.1 General case—random graphs

The first set of experiments were performed using random graphs as depicted in Figure 7a. However, we vary the number of workers according to the previously described configurations.

Table 3 shows the numerical results, which are visualized in the previous Figure 8.

7.3.2 Specific case—preferential attachment graphs

The second set of experiments were performed using preferential attachment graphs as depicted in Figure 7b. Again, we vary the number of workers according to the previously described configurations.

The Table 4 shows the numerical results, which are visualized in Figure 9.

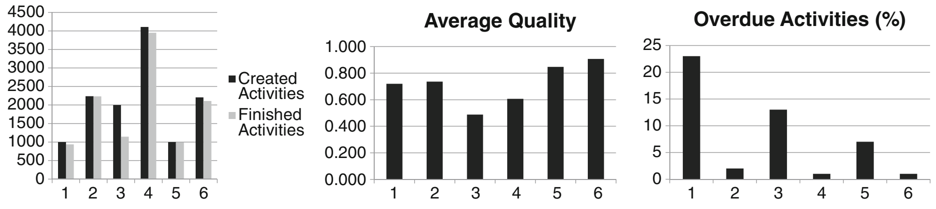
7.4 Overall findings

Both sets of figures, Figures 8 and 9 show the results of our experiments by comparing the different pairs of configurations. The horizontal axis of each figure shows the index of a configuration that corresponds to the simulation parameters as defined in Table 2. In general, both graphs (random and preferential attachment) exhibit similar results with only minor differences. This means that our proposed ranking approach is applicable to both, sparsely connected random graphs as well as more densely connected preferential attachment graphs. Thus, the following discussions apply to both sets of experiments using respective graph structure.

The first series of experiments shows the relation of the number of created activities versus the number of finished activities. Without advanced processing, an activity is simply created based on the properties of a human tasks and assigned to individual workers. On the other hand using advanced processing, if the duration of a task exceeds a certain duration threshold, an activity is created that is split into

Table 3 Numerical values of experiment results using random graph.

Configuration	1	2	3	4	5	6
Created activities	1000	2237	2000	4106	1000	2208
Finished activities	940	2234	1147	3950	989	2108
Average quality	0.720	0.736	0.488	0.607	0.847	0.907
Overdue activities (%)	23	2	13	1	7	1



(a) Activity creation. (b) Activity quality. (c) Activities overdue.

Figure 8 Experiment results using random graph.

multiple sub-activities. The supervisors distributes sub-activities in the context of a parent activity, assembles the result, and passes it on to the coordinator.

Both Figures 8a and 9a show that the number of activities is always higher in social-crowd environments (i.e., advanced processing) because activities are split and reassigned to workers. However, the number of finished activities in relation to the number of created activities is always higher when compared to the regular processing behavior. This means that advanced processing increases the number of created *and* successfully finished activities (i.e., the reliability in processing activities in crowdsourcing environments increases).

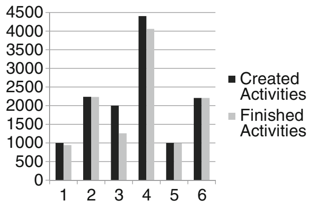
Figures 8b and 9b visualize the average quality obtain in different experiment configurations. The quality of a task result is based on the worker's skill (regular processing) or the supervisor's skill (advanced processing). Thus, in the latter case the quality is ensured by the supervisor. The average quality of tasks is always higher in the advanced processing case. This is the result of our ranking approach which ensures that coordinators are ranked higher if they are connected to skilled supervisors. Comparing the pairs of configurations, the quality in the configuration pair 3 and 4 is lower due to the larger number of activities to be processed. However, our advanced processing approach still outperforms the regular processing setting in terms of providing better quality results. Also, given a larger social network of 200 workers the task quality is higher.

Finally, Figures 8c and 9c show the number of overdue activities which were not processed on time (deadline violations). The percentage ratio of overdue activities is much lower in the social-crowd environment because larger tasks (based on effort/duration of a task) are split into smaller crowd-activities which are processed faster than larger chunks of work. It is easier to assign smaller tasks to crowd members instead of finding people to process larger tasks; thereby reducing the number of overdue activities.

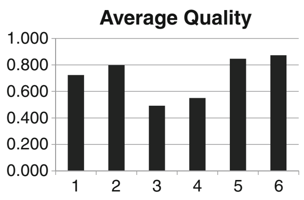
To conclude our discussions, we confirm that the proposed social-crowd environment has a number of advantages over traditional environments that are based on a

Table 4 Numerical values of experiment results using preferential attachment graph.

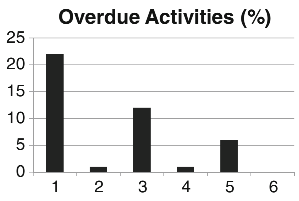
Configuration	1	2	3	4	5	6
Created activities	1000	2237	2000	4406	1000	2208
Finished activities	944	2233	1258	4062	989	2208
Average quality	0.724	0.799	0.492	0.550	0.847	0.873
Overdue activities (%)	22	1	12	1	6	0



(a) Activity creation.



(b) Activity quality.



(c) Activities overdue.

Figure 9 Experiment results using preferential attachment graph.

population of workers which perform tasks separately. Our experiments show that task quality is increased while improving reliability and performance of the crowd.

8 Conclusion

Crowdsourcing has emerged as an important paradigm in human problem solving techniques on the Web. In such environments, people offer their skills and capabilities in a service-oriented manner. However, one cannot rely on the constant availability of people. The dynamic discovery of skilled people becomes a key aspects. Here we proposed social-crowds that collaboratively process tasks. We designed extensions for BPEL4People to utilize crowds in process-centric enterprise environments. We explained in detail various extensions to cope with quality issues. Furthermore, we proposed a role detection algorithm to build up hierarchical social networks. The presented social-crowd environment brings a number of benefits including (1) increased task quality and (2) an increased number of successfully finished activities as well as (3) a reduced number of overdue activities. We believe that social-crowd environments have a great potential to make crowdsourcing more reliable while increasing quality of task results.

Task costs in crowdsourcing have not been detailed in this work (see our previous work in [37, 40]) but will be addressed in the context of B4P in future work. Also, we plan to utilize Mechanical Turk for experiments with real people. Also, we will investigate the integration of various XML-based standards and interfaces including B4P, WS-HT, and MTurk's API.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable comments which helped to improve the quality of this paper. Furthermore, we would like to thank Frank Leymann for discussions on extending Web service standards including WS-HumanTask and BPEL4People.

References

1. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: a service-oriented implementation of dynamic flexibility in workflows. In: OTM Conferences (1), pp. 291–308 (2006)
2. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: WSDM, pp. 183–194. ACM (2008)
3. Agrawal, A., et al.: Ws-bpel extension for people (bpel4people), version 1.0. (2007)

4. Amend, M., et al.: Web services human task (ws-humantask), version 1.0. (2007)
5. Andrews, T., et al.: Business process execution language for web services, version 1.1. (2003)
6. Balthazard, P.A., Potter, R.E., Warren, J.: Expertise, extraversion and group interaction styles as performance indicators in virtual teams: how do perceptions of it's performance get formed? Data Base **35**(1), 41–64 (2004)
7. Barabasi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
8. Benkler, Y.: Coase's penguin, or linux and the nature of the firm. CoRR, cs.CY/0109077 (2001)
9. Brabham, D.: Crowdsourcing as a model for problem solving: an introduction and cases. Convergence **14**(1), 75 (2008)
10. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **25**, 163–177 (2001)
11. Breslin, J., Passant, A., Decker, S.: Social web applications in enterprise. Social Semantic Web **48**, 251–267 (2009)
12. Cozzi, A., Farrell, S., Lau, T., Smith, B.A., Drews, C., Lin, J., Stachel, B., Moran, T.P.: Activity management as a web service. IBM Syst. J. **45**(4), 695–712 (2006)
13. Cugola, G., Nitto, E.D., Fuggetta, A., Ghezzi, C.: A framework for formalizing inconsistencies and deviations in human-centered systems. ACM Trans. Softw. Eng. Methodol. **5**(3), 191–230 (1996)
14. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the World-Wide Web. Commun. ACM **54**(4), 86–96 (2011). doi:10.1145/1924421.1924442
15. Dolev, S., Elovici, Y., Puzis, R.: Routing betweenness centrality. J. ACM **57**, 25:1–25:27 (2010)
16. Dom, B., Eiron, I., Cozzi, A., Zhang, Y.: Graph-based ranking algorithms for e-mail expertise analysis. In: DMKD, pp. 42–48. ACM (2003)
17. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press (2010)
18. Gentry, C., Ramzan, Z., Stubblebine, S.: Secure distributed human computation. In: EC '05, pp. 155–164. ACM (2005)
19. Herrmann, K., Rothermel, K., Kortuem, G., Dulay, N.: Adaptable pervasive flows—an emerging technology for pervasive adaptation. In: Workshop on Pervasive Adaptation (PerAda) (2008)
20. Howe, J.: The rise of crowdsourcing. http://www.wired.com/wired/archive/14.06/crowds.html (2006)
21. IBM: An architectural blueprint for autonomic computing (whitepaper) (2005)
22. Ipeirotis, P.G.: Analyzing the Amazon mechanical turk marketplace. SSRN eLibrary **17**(2), 16–21 (2010)
23. Kleinberg, J.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)
24. Kleinberg, J.: The convergence of social and technological networks. Commun. ACM **51**(11), 66–72 (2008)
25. Kosorukoff, A., Goldberg, D.E.: Genetic algorithms for social innovation and creativity. Technical report, University of Illinois at Urbana-Champaign (2001)
26. Kumar, A., W.M.Aalst, P.V.D., Verbeek, E.: Dynamic work distribution in workflow management systems: how to balance quality and performance. J. Manage Inf. Syst. **18**(3), 157–193 (2002)
27. Lampe, C., Resnick, P.: Slash(dot) and burn: distributed moderation in a large online conversation space. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, pp. 543–550. ACM, New York, NY, USA (2004)
28. Liu, L., Thanheiser, S., Schmeck, H.: A reference architecture for self-organizing service-oriented computing. In: ARCS, pp. 205–219 (2008)
29. Maximilien, E.M., Singh, M.P.: Toward autonomic web services trust and selection. In: ICSOC '04, pp. 212–221. ACM (2004)
30. Mendling, J., Ploesser, K., Strembeck, M.: Specifying separation of duty constraints in bpel4people processes. In: BIS'08, pp. 273–284. Springer Verlag (2008)
31. Newman, M.E., Strogatz, S.H., Watts, D.J.: Random graphs with arbitrary degree distributions and their applications. Phys. Rev. E **64**(5), 026118 (2001). doi:10.1103/PhysRevE.64.026118
32. Newman, M.E.J.: The structure of scientific collaboration networks. Proc. Natl. Acad. Sci. **98**, 404–409 (2001)
33. Newman, M.E.J., Watts, D.J., Strogatz, S.H.: Random graph models of social networks. Proc. Natl. Acad. Sci. U.S.A. **99**(Suppl 1), 2566–2572 (2002)
34. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
35. Panteli, N., Davison, R.: The role of subgroups in the communication patterns of global virtual teams. IEEE Trans. Prof. Commun. **48**(2), 191–200 (2005)

36. Petrie, C.: Plenty of room outside the firm. Internet Comput. **14**, 92–96 (2010)
37. Psaier, H., Skopik, F., Schall, D., Dustdar, S.: Resource and agreement management in dynamic crowdcomputing environments. In: EDOC (2011)
38. Reka, A., Barabási, A.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (2002)
39. Russell, N., W.M.Aalst, P.V.D.: Evaluation of the bpel4people and ws-humantask extensions to ws-bpel 2.0 using the workflow resource patterns. Technical report, BPM Center Brisbane/Eindhoven (2007)
40. Satzger, B., Psaier, H., Schall, D., Dustdar, S.: Stimulating skill evolution in market-based crowdsourcing. In: BPM. Lecture Notes in Computer Science. Springer (2011)
41. Schall, D.: Human interactions in mixed systems—architecture, protocols, and algorithms. PhD thesis, Vienna University of Technology (2009)
42. Schall, D.: A human-centric runtime framework for mixed service-oriented systems. Distrib. Parallel Databases **29**(5–6), 333–360 (2011). doi:10.1007/s10619-011-7081-z
43. Schall, D.: Expertise ranking using activity and contextual link measures. Data Knowl. Eng **71**(1), 92–113 (2012)
44. Schall, D., Skopik, F., Dustdar, S.: Expert discovery and interactions in mixed service-oriented systems. IEEE Trans. Services Comput. **5**, 233–245 (2012). doi:10.1109/TSC.2011.2
45. Schall, D., Truong, H.-L., Dustdar, S.: Unifying human and software services in web-scale collaborations. IEEE Internet Comput. **12**(3), 62–68 (2008)
46. Shetty, J., Adibi, J.: Discovering important nodes through graph entropy the case of enron email database. In: LinkKDD, pp. 74–81. ACM (2005)
47. Shi, X., Bonner, M., Adamic, L.A., Gilbert, A.C.: The very small world of the well-connected. In: HT '08, pp. 61–70. ACM (2008)
48. Siorpaes, K., Simperl, E.: Human intelligence in the process of semantic content creation. World Wide Web **13**, 33–59 (2010). doi:10.1007/s11280-009-0078-0
49. Skopik, F., Schall, D., Dustdar, S.: Modeling and mining of dynamic trust in complex service-oriented systems. Inf. Syst. **35**, 735–757 (2010)
50. Su, Q., Pavlov, D., Chow, J.-H., Baker, W.C.: Internet-scale collection of human-reviewed data. In: WWW '07, pp. 231–240. ACM (2007)
51. Thomas, J., Paci, F., Bertino, E., Eugster, P.: User tasks and access control over web services. In: ICWS '07, pp. 60–69. IEEE (2007)
52. von Ahn, L.: Games with a purpose. IEEE Comput. **39**(6), 92–94 (2006)
53. Vukovic, M.: Crowdsourcing for enterprises. In: Proceedings of the 2009 Congress on Services, pp. 686–692. IEEE Computer Society (2009)
54. Yang, J., Adamic, L., Ackerman, M.: Competing to share expertise: the taskcn knowledge sharing community. In: International Conference on Weblogs and Social Media (2008)
55. Zhang, J., Ackerman, M.S., Adamic, L.: Expertise networks in online communities: structure and algorithms. In: WWW, pp. 221–230. ACM (2007)
56. Zhao, X., Liu, C., Sadiq, W., Kowalkiewicz, M., Yongchareon, S.: Implementing process views in the web service environment. World Wide Web **14**(1) 27–52 (2011)