QoS-Based Task Scheduling in Crowdsourcing Environments*

Roman Khazankin, Harald Psaier, Daniel Schall, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology,
Argentinierstrasse 8/184-1, A-1040 Vienna, Austria
{lastname}@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at

Abstract. Crowdsourcing has emerged as an important paradigm in human-problem solving techniques on the Web. One application of crowdsourcing is to outsource certain tasks to the crowd that are difficult to implement as solutions based on software services only. Another benefit of crowdsourcing is the on-demand allocation of a flexible workforce. Businesses may outsource certain tasks to the crowd based on workload variations. The paper addresses the monitoring of crowd members' characteristics and the effective use of monitored data to improve the quality of work. Here we propose the extensions of standards such as Web Service Level Agreement (WSLA) to settle quality guarantees between crowd consumers and the crowdsourcing platform. Based on negotiated agreements, we provide a skill-based crowd scheduling algorithm. We evaluate our approach through simulations.

Keywords: crowdsourcing, skill monitoring, scheduling, QoS agreements.

1 Introduction

Recently, business processes need to be adapted or extended more frequently to the changing market. Companies often lack the new capabilities or knowledge required. To tackle those changes, either new personal needs to be hired, or rather, the new process steps are outsourced. The work in this paper is based around a recent and attractive type of outsourcing called *crowdsourcing* [11]. The term crowdsourcing describes a new web-based business model that harnesses the creative solutions of a distributed network of individuals [4], [21]. 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, LiveOps, or Yahoo! ([3], [17], [22]) have recognized the opportunities behind such *mass collaboration systems* [8] for both

* This work was supported by the European Union FP7 projects COIN (No. 216256) and SCube (No. 215483) and the Vienna Science and Technology Fund (WWTF), project ICT08-032.

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 297–311, 2011.

improving their own services and as a business case. The most prominent platform they currently offer is the *Amazon Mechanical Turk* (AMT) [3]. Requesters are invited to issue *human-intelligence tasks* (HITs) requiring a certain qualification to the AMT. The registered customers post mostly tasks with minor effort that, however, require human capabilities (e.g., transcription, classification, or categorization tasks [12]).

In this paper we extend this simple model and focus on crowdsourcing platforms that deal with task groups consisting of manifold similar jobs provided by consumers. Most current public crowdsourcing platforms with market-like operation chain announce received tasks at their portal as a first step. Next, the worker chooses among the assorted mass of task those s/he likes to process. The selection is motivated by her/his personal preferences. Thus, the following assignment is initiated by the worker, and as a consequence, it hardly allows the system to have an influence upon assignments and to leverage the skill heterogeneity of involved workers.

As each worker is individual, it is natural that the skills of the workers are manifold. The tasks submitted to the platform are also diverse in their requirements. Hence, efficient crowdsourcing must consider the suitability of a worker for a task. One can assume that the more the worker is suitable for a task, the better the expected outcome quality is. Therefore, given the suitability information and a control mechanism for a worker assignment, it is possible to improve the overall result's quality by assigning tasks to best suitable workers for the current situation. Moreover, from a BPM perspective, this mechanism also provides control over task completion times which can be used to maintain objectives, such as deadline fulfillment. To sum up, the task assignment control would enable a crowdsourcing platform provider to develop QoS policies for offered crowdsourcing services, so these services can be integrated into QoS-sensitive business processes. The assignment control demands for a scheduling problem to be solved. The problem is to maximize overall quality while satisfying agreed objectives. Such a scheduling problem is hindered by a number of crowd-specific features, such as lack of full control of the workers and their membership, and their limited predictable availability.

In this paper we tackle these issues by proposing crowdsourcing platform enhancements. Hence, our key contributions are:

- A crowdsourcing platform model which allows for agreement-aware task processing.
- Algorithms for task scheduling and workers' skill profile updates.
- Proof-of-concept implementation and evaluation of the approach in a simulated environment.

The paper is organized as follows. In Sect. 2 we list work related to crowdsourcing and scheduling. Section 3 outlines our platform model that provides agreement-based task assignments to a crowd and also gives insights in the application of current crowdsourcing platforms. Section 4 details the proposed crowdsourcing model. A prototype of the platform is evaluated in Sect. 5. Section 6 concludes the work.

2 Related Work

In this work we position crowdsourcing in a service-oriented business setting by providing automation. In crowdsourcing environments, people offer their skills and capabilities in a service-oriented manner. Major industry players have been working towards standardized protocols and languages for interfacing with people in SOA. Specifications such as WS-HumanTask [10] and BPEL4People [2] have been defined to address the lack of human interactions in service-oriented businesses [16]. These standards, however, have been designed to model interactions in closed enterprise environments where people have predefined, mostly static, roles and responsibilities. Here we address the service-oriented integration of human capabilities situated in a much more dynamic environment where the availability of people is under constant flux and change [6]. The recent trend towards *collective intelligence* and crowdsourcing can be observed by looking at the success of various Web-based platforms that have attracted a huge number of users. Well known representatives of crowdsourcing platforms are the aforementioned form Yahoo!, LiveOps, and Amazon. The difference between these platforms lies in how the labor of the crowd is used.

Crowdsourcing. AMT, for example, offers access to a large number of crowd workers. With their notion of HITs that can be created using a Web service-based interface they are closely related to our aim of mediating the capabilities of crowds to service-oriented business environments. Despite the fact that AMT offers HITs on various topics [12], the major challenges are to find on request skilled workers that are able to provide high quality results for a particular topic (e.g., see [1]), to avoid spamming, and to recognize low-performers. To the best of our knowledge, these problems are still not faced by AMT. In this work we focus on those issues.

Another shortcoming of most existing real platforms is the lack of different and comprehensive *skill information*. Most platforms have a simple measure to prevent workers (in AMT, a threshold of task success rate can be defined) from claiming tasks. In [19], the automated calculation of expertise profiles and skills based on interactions in collaborative networks was discussed.

In [13], a quality management approach for crowdsourcing environments is presented. Unlike our profile management, this work doesn't support multiple skills, but concentrates on a single correctness dimension. On the other hand, if there is a specific need for such a quality management technique, the profile management can thus be replaced with it by correllating the correctness and suitability, as this module is decoupled from the rest of the platform as mentioned in Sec. 3.

Scheduling is a well-known subject in computer science. The novel contribution in this work is to consider multidimensional assignment and allocation of tasks. A thorough analysis and investigation in the area of multidimensional optimal auctions and the design of optimal scoring rules has been done by [7]. In [18] iterative multi-attribute procurement auctions are introduced while focusing on

mechanism design issues and on solving the multi-attribute allocation problem. Focusing on task-based adaptation, [20] near-optimal resource allocations and reallocations of human tasks were presented. Staff scheduling related to closed systems was discussed in [5,9]. However, unlike in closed enterprise systems, crucial scheduling information, i.e., the current user load or precise working hours are usually not directly provided by the crowd. Instead, the scheduling relevant information must be gathered by monitoring. The work in [15] details the challenges for collaborative workforce in crowdsourcing where activities are coordinated, workforce contributions are not wasted, and results are guaranteed.

3 Crowdsourcing Platform Model

In this section a model of a crowdsourcing platform (see Fig. 1) which supports agreement-based task assignment is outlined. To begin with, there is a negotiation that states the objectives between the consumer and the platform regarding the coming task assignments. The life-cycle of a task begins at the consumer. S/He submits a collection of tasks to the crowd. We assume that the tasks have distinct skill requirements, thus, need to be assigned accordingly. After processing, the result is returned to the consumer which is invited to provide a quality feedback on the result.

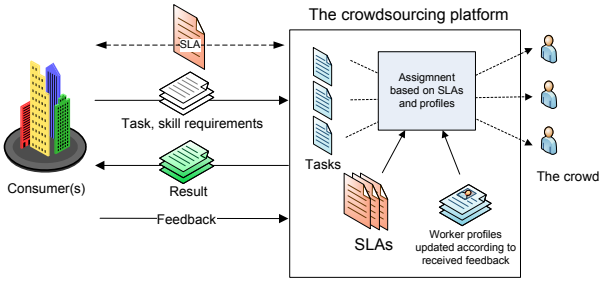


Fig. 1. Platform model

Once a consumer submits a task, s/he provides skill requirements along with the task. In the assignment phase the platform estimates the matching between tasks and workers. The more a worker's skills match the requirements the more this worker is suitable for the task. Also, in service-oriented environments a *Service Level Agreement (SLA)* usually is negotiated which has influence on the assignment strategy. An SLA includes temporal and quality requirements, and preferences. The challenge for the platform is to negotiate the *Service Level Objectives (SLOs)* of an SLA related to possible assignments. These SLOs must base on the observed and predicted behavior of the crowd and the current situation. In particular, already distributed tasks and the resulting task load must be considered.

Therefore, active tasks are assigned not only according to suitability of workers, but also, in line with the SLOs of the agreement. The agreement aims to

avoid or minimize the losses and to maximize the overall result's quality. In other words, the objective is to maximize the quality while enforcing the SLA. Also, the availability of workers is taken into consideration by requesting short-term information regarding their ability to perform tasks.

The suitability is calculated as a match between required skills for the task and the skills of a worker. The skills of crowd workers are maintained in their profiles by the platform management. Initially, skill information is provided by the workers themselves.

However, this information must not be considered complete and reliable. Some workers might not know about their real skill-levels or overestimate their capabilities. Hence, the platform management must be allowed to monitor the activities in the crowd and update the created profiles according to the observations. The resulting real quality is reported by the consumer's feedback. The difference between expected quality and required skills for the tasks hint the real skills of the workers. Also, if an assignment is refused by a worker, despite his claim for availability, various penalty sanctions can be imposed to this worker.

The rules for calculating the suitability are independent of the scheduling logic. Thus, the suitability is represented by a single real value in $[0, 1]$ (0 - not suitable at all, 1 - perfectly suitable) which summarizes the expectations regarding the quality of the result. This enables the technique, which is used to calculate the suitability, to be completely decoupled from scheduling.

3.1 Integration of Service Level Agreements (SLAs)

Assignment control enables the platform to estimate the crowd occupancy and to give certain guarantees to consumers. Such guarantees can be given in form of SLAs, and allow to integrate crowdsourcing activities in service-oriented environments. As SLAs are crucial for business process management, such a model can substantially sustain the use of crowdsourced services in business processes.

Next, we discuss an outline of a WSLA[1] document that could be exchanged and agreed between consumer and crowd platform.

After the contract parties' details, *SchedulingPlatform* and *Consumer* listed in lines 4 to 11, Listing 1.1 states the contract items from line 12 to 20. These are a collection of *ServiceObjectType* items including scheduling, operation description, and configuration, and also, the *SLAParameter*s. Here, as an example the parameter (`TaskSkills`) uses a crowd particular metric `CompareSkills` to compare skill profiles of the workers to the skill required in the document. Important to note, we extend the WSLA definition with our own namespace (task scheduling platform `tsp`) defining xpath methods for expressions and type definitions to comply with all requirements of the platform.

Listing 1.2 shows the agreement's terms as *Obligations* of the contract including some SLOs. An SLO consists of an obliged party, a validity period, and an *Expression*s that can be combined with a logic expression (e.g., *Or*). The *Value* tag in the predicates of WSLA is restricted to double. Hence, another extension

[1] http://www.research.ibm.com/wsla/

```
1  <wsla:SLA
2  xmlns:wsla="http://www.ibm.com/wsla" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xmlns:tsp="http://www.infosys.tuwien.ac.at/tsp/" name="SLA4711">
4   <wsla:Parties>
5    <wsla:ServiceProvider name="SchedulingPlatform">
6     <!-- details -->
7    </wsla:ServiceProvider>
8    <wsla:ServiceConsumer name="Consumer">
9     <!-- details -->
10   </wsla:ServiceConsumer>
11  </wsla:Parties>
12  <wsla:ServiceDefinition name="CrowdService">
13   <wsla:Operation xsi:type="wsla:WSDLSOAPOperationDescriptionType" name="ScheduleTask">
14    <!-- schedule period -->
15    <wsla:SLAParameter name="TaskSkills" type="tsp:SkillList" unit="double">
16     <wsla:Metric>CompareSkills</wsla:Metric>
17    </wsla:SLAParameter>
18    <!-- SLAParameter metrics for NrOfTasks, TaskQuality, Fee. Details: name, wsdl-location, binding -->
19   </wsla:Operation>
20  </wsla:ServiceDefinition>
21  <!-- wsla Obligations -->
22 </wsla:SLA>
```

Listing 1.1. Involved parties and body

with xpath methods allows us to provide more complex expressions for the values (e.g., see Line 8). Generally, in an SLO an evaluation event defines the trigger for the evaluation of the metric function. The content of the expressions connects the pool of *SLAParameters* of the items to a predicate (e.g, `GreaterEqual`) and threshold value (*Value*). In the example we define three objectives. The first, `sloSkills` defines that the match between the task skill requirement `skills required` and the selected potential worker's skills must be greater or equal. The second, `sloQuality` is a composed objective by an *Or* expression. The agreement states, that either a defined number of tasks (`100`) is delivered at the end of an agreed interval (e.g., *wsdl:Months*) or the focus of processing is on the task's quality, and hence, a result quality of at least 80% is expected. The final objective `sloFee` obliges the consumer to confirm the quality of the result and pay the related fee. In the example, similar to Line 8, the xpath method `getInput` parses a document `TaskDesc` and returns the task's fee. As the fee depends also on the result, the result report `TaskRes` contains the reported quality. Multiplied they give the final fee due.

3.2 Discussion

Generally, in our model, workers are more restricted than in market-oriented crowdsourcing platforms. Still, such an approach provides adequate flexibility for workers by allowing them to choose the time periods in which they are willing to work. Thus, assuming a strong competition among workers, the model will be feasible. Constantly increasing interest in crowdsourcing platforms [8] indicates that such competition is quite realistic. Also, SLA-enabled services are higher valued, therefore, the monetary compensation, and, thus, the competition can be higher. The feedback provision from the consumer is of his/her own interest,

```
<wsla:Obligations>
 <wsla:ServiceLevelObjective name="sloSkills" serviceObject="ScheduleTask">
  <wsla:Obliged>SchedulingPlatform</wsla:Obliged>
  <!-- Validity -->
   <wsla:Expression> <!-- skill requirements -->
    <wsla:Predicate xsi:type="wsla:GreaterEqual">
     <wsla:SLAParameter>TaskSkills</wsla:SLAParameter>
     <tsp:Value>tsp:getInput("TaskDesc")//TaskDefinition/skills_required</tsp:Value>
    </wsla:Predicate>
   </wsla:Expression>
  <wsla:EvaluationEvent>TaskAssignment</wsla:EvaluationEvent>
 </wsla:ServiceLevelObjective>
 <wsla:ServiceLevelObjective name="sloQuality" serviceObject="ScheduleTask">
  <wsla:Obliged>SchedulingPlatform</wsla:Obliged>
  <!-- Validity -->
  <wsla:Or>
  <wsla:Expression>
   <wsla:Predicate xsi:type="wsla:Equal">
    <wsla:SLAParameter>NrOfTasks</wsla:SLAParameter>
    <wsla:Value>100.0</wsla:Value>
   </wsla:Predicate>
  </wsla:Expression>
  <wsla:Expression>
   <wsla:Predicate xsi:type="wsla:GreaterEqual">
    <wsla:SLAParameter>TaskQuality</wsla:SLAParameter>
    <wsla:Value>0.8</wsla:Value> <!-- expected qty 80%-->
   </wsla:Predicate>
  </wsla:Expression>
  </wsla:Or>
  <wsla:EvaluationEvent>TaskResult</wsla:EvaluationEvent>
 </wsla:ServiceLevelObjective>
 <wsla:ServiceLevelObjective name="sloFee" serviceObject="ScheduleTask">
  <wsla:Expression>
  <wsla:Obliged>Customer</wsla:Obliged>
   <wsla:Predicate xsi:type="wsla:Equal">
    <wsla:SLAParameter>Fee</wsla:SLAParameter>
    <tsp:Value>
     <![CDATA[tsp:getInput("TaskDesc")//Fee * tsp:getInput("TaskRes")//Quality]]>
        </tsp:Value>
   </wsla:Predicate>
  </wsla:Expression>
  <wsla:EvaluationEvent>TaskResult</wsla:EvaluationEvent>
 </wsla:ServiceLevelObjective>
       <!-- agreed qualified actions on slo violations -->
</wsla:Obligations>
```

Listing 1.2. Obligations and SLOs

as it positively affects the result quality. It can be provided not for all results but selectively.

In this paper we don't discuss the cost of the work and payments in detail. Although it is an important factor, in our vision, it can be seamlessy integrated into the platform. One design solution could be that the workers specify the minimal cost for their work and the consumers would pay as much as they want as in a traditional crowdsourcing platform. Thus, the more the customer is willing to pay, the more workers would be considered for assignment, and, as it is sensibly to assume, more suitable workers could be found. However, such a design will not change the basics and the algorithms of our platform substantially. Thus, for the sake of simplicity, we assume that all the jobs cost correspondingly to their specified duration.

4 Quality and Skill-Aware Crowdsourcing

This section explains in detail the quality and skill-aware crowdsourcing platform architecture, which is a proof-of-concept implementation of the model described in Sect. 3.

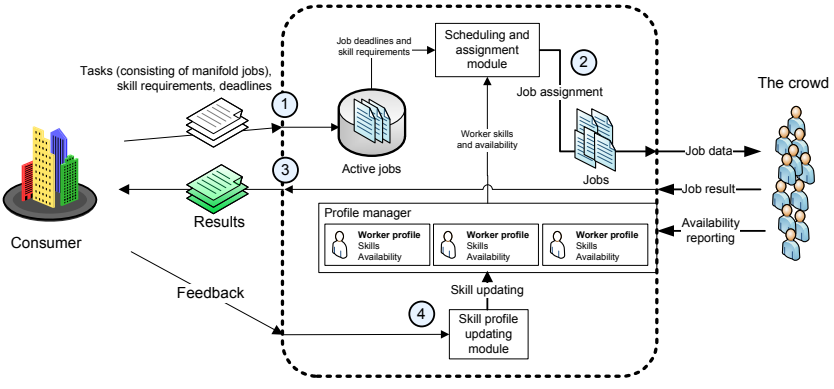


Fig. 2. Platform architecture

The platform behavior can be summarized by the following steps (in the line with the ones in Fig. 2):

1. A consumer submits a task that consists of manifold similar jobs. The consumer also specifies required skills and the deadline for the task, so all the jobs should be processed until this deadline. The task is added to active task pool.
2. The scheduling module periodically assigns active jobs to workers according to deadlines and suitability of workers.
3. When a job is done, the result is sent to the consumer.
4. The consumer can provide a feedback about the result quality. It is reported to the skill profile updating module, which matches it with the expected quality and corrects the worker skill profile if necessary.

Of course, the platform is intended for usage by multiple consumers, this fact is not depicted for simplicity. A deterministic time model is used in the platform, so the time is discreet and is represented by sequential equally long time periods. A time period can represent any time duration like a minute, an hour, or a day.

4.1 Skills and Suitability

The system distinguishes a fixed set of skills. Each worker has a skill profile, where each skill is described by a real number $s \in [0, 1]$ which defines it quantitatively (0 - lack of skill, 1 - perfect).

Each submitted task has the required skills specified. Each required skill is also represented as a real number $r \in [0, 1]$. If $r = 0$ then the quality does not depend on this skill. If $r > 0$ then the best outcome quality is expected in case if the corresponding worker's skill s is $s >= r$. If $s < r$ then the expected quality is affected in the inverse proportion to $r - s$. The quality is again represented as a real number $q \in [0, 1]$. The suitability of a worker for a task is equal to the expected outcome quality. The exact matching formula is shown below.

Let WS_i - worker skills, RS_i - required skills of a task, $i = \overline{1, N}, N-$ number of skills. Then the suitability of the worker to the task is calculated as

$$S = 1 - \sum_{i \in M} \frac{Max((RS_i - WS_i)/RS_i, 0)}{|M|} \qquad M : k \in M \Leftrightarrow k \in N, RS_k > 0$$

Thus, the more worker's skills are proportionally closer to the required skills of a task, the more the worker is suitable to the task. If the worker's skill is equal or greater than the corresponding required skill, then this skill suits perfectly.

4.2 Worker and Consumer Communication

As the user interface is not the focus of our work, the communication with consumers and workers is performed by means of web services. The full implementation of the platform can employ, e.g., a web interface developed on top of these web services.

4.3 Scheduling

Scheduling is performed based on deadlines and skill requirements also derived from an SLA. The objective of the scheduling module is to maximize the overall quality, while fulfilling deadlines. The assumption is that missing a deadline can not be justified by any quality gain, thus, meeting a deadline is the first-priority objective, and the quality maximization is the second-priority objective.

Algorithm 1 describes a scheduling algorithm which is used in our platform. The idea behind the algorithm is that the best quality is achieved when a task is assigned to most suitable workers. The quality is higher when a task is performed by a smaller number of best workers, but this number should not be too small, so the task can be finished until the deadline. This number is calculated in $toTake$ for each active task. The tasks with earlier deadlines are assigned in the first place. In an attempt to improve the algorithm's efficiency, we tried a number of heuristic extensions, such as:

- Based on reported short-time worker availability, assign less jobs at a given time to wait for more suitable workers to become available (while avoiding possible crowd "overloads")
- Assign more jobs at a given time if the suitability of additional workers is almost as good as the suitability of best workers.
- Having $toTake$ numbers calculated, optimize the worker-task assignments for each time period using an optimization framework.

Algorithm 1. Greedy scheduling algorithm.

Require: $currentTime$ current time
Require: $tasks$ active tasks
1: **for** $task \in tasks$ in the order of ascending $task.deadline$ **do**
2: $stepsToDeadline = (task.deadline - currentTime+1)$ / $task.duration$ - 1
3: **if** $stepsToDeadline > 0$ **then**
4: **if** $(task.deadline - currentTime + 1)$ % $task.duration) > 0$ **then**
5: $toTake = 0$
6: **else**
7: $toTake = Trunc(task.numberOfJobsToDo/stepsToDeadline)$
8: **end if**
9: **else**
10: $toTake = task.numberOfJobsToDo$
11: **end if**
12: **while** $toTake > 0$ AND some workers are still available **do**
13: Assign a job of $task$ to most suitable available worker for $task$
14: $toTake = toTake - 1$
15: **end while**
16: **end for**

However, as shown in Sect. 5, such extensions do not give a substantial improvement. We believe that the reason of such a weak improvement is the size of the crowd: if a worker cannot be assigned to a due task, in most of the cases a good enough replacement for the worker can be found. Thus, we conclude that a greedy algorithm is generally sufficient for scheduling in a crowdsourcing environment. The refinement of the algorithm can be done according to the particular crowd characteristics that can be estimated only when the system is used by real users in the commercial operation.

4.4 Profile Management

The crowd workers' profile management is of major importance in our assumptions. As mentioned before, the success of the scheduling algorithm partially depends on the accuracy of the profile monitoring. At the beginning of her/his membership at the crowdsourcing platform a user registers with a profile representing the skills. Usually this information is not very accurate because users tend to over-/underestimate their skills. Hence at runtime, a monitoring module must run on-line and manage the profiles by updating the provided information. It is necessary to avoid conflicts with the promised quality agreements (c.f., Sect. 3). This is a major challenge. The task processing results and the expected quality outcome must be used as a reference for the real skills of a worker. The quality expectations on the tasks result are often detailed in the task description. At the AMT, for example, the result feedback contains usually only a task accept or reject. At our platform, with an agreement requiring the customer to give a feedback on the quality, the feedback contains crucial information for the Algorithm 2 that can be used to update the skills of the reported worker profiles.

As the scheduler requires skill knowledge the profile update is twofold. If the worker only provided a low quality the update depends on the difference (Lines

Algorithm 2. Profile monitoring.

Require: QF quality feedback of the provider, QE quality expected by the provider
Require: $worker$ processing worker and $taskSkills$ required task skills
1: $workerSkills \leftarrow worker.getSkills()$
2: **if** $QE > \vartheta_q$ /* high quality result */ **then**
3: /* compare with latest history entry, update and continue on better QF */
4: $entry \leftarrow setHistory(QF, taskSkills)$
5: **for** skill $s \in workerSkills$ **do**
6: $reqSkill \leftarrow getTaskSkills(s)$
7: $diff \leftarrow |s - reqSkill| \times \alpha_w$
8: **if** $s > reqSkill$ **then**
9: $workerSkills.set(s + diff)$
10: **else**
11: $workerSkills.set(s - diff)$
12: **end if**
13: **end for**
14: **return**
15: **end if**
16: /* low quality result */
17: $wprofile \leftarrow setOfProfiles.get(worker)$ /* set of registered profiles */
18: $diff \leftarrow QF/QE$ /* difference between the qualities */
19: **for** skill $s \in workerSkills$ **do**
20: /* skill == 1 perfect knowledge */
21: **if** $skill \times diff <= 1$ **then**
22: $workerSkills.set(s \times diff)$
23: **end if**
24: **end for**

17-24). If the quality is above a certain threshold ϑ_q and is better than a previous then we consider the required skills close to the workers own (Line 3-14). Hence, the difference between the required and the worker's own skills (weighed by the factor α_w) influence the worker's skill update.

5 Experiments

To evaluate our platform, we set up a simulated environment that comprises a crowd which perform tasks and consumers who submit tasks and provide the feedback. We assigned a real skill set for each simulated worker to calculate the real quality outcome (which consumers report) using the suitability formula (see Sect. 4.1). The platform management had no access to the real skills, but was only able to estimate workers' skills based on the feedback provided by consumers. We evaluated the average job quality in different setups, varying the workload of the crowd, the availability of workers, the scheduling algorithms, and the skill awareness.

Simulation of a real crowdsourcing environment is challenging due to the lack of comprehensive statistical data in this area. Although we don't rely on any real data in our simulations, we tried our best to prognosticate the meaningful simulation parameters based on our knowledge and experience.

5.1 Experiment Setup

In our experiments we use a set of 10 skills for describing worker skills and task skill requirements.

Customers. The customers submit tasks to the platform and provide the feedback on completed jobs. Tasks are submitted randomly while ensuring the average crowd workload and avoiding overloads.

Each task comprises skill requirements, number of jobs, and deadline. During each time period of the simulation, if the *Task Limit* has not been reached yet, a new task is submitted to the system with *Task Concentration* probability. The job duration is calculated as $Min(1 + abs(\phi/2 * \sigma), \sigma + 1)$, where ϕ is a normally distributed random value with mean 0 and standard deviation 1. The deadline is assigned randomly according to *Steps To Deadline* parameter. The number of jobs is calculated so that the crowd workload is near equally distributed among the tasks, and the average workload remains close to *Intended Schedule Density*. The parameters and their values are described in Table 1.

Table 1. Task generation parameters

Name	Description	Value(s)
Tasks Limit	The total number of submitted tasks	200
Job Duration Sigma (σ)	Describes the deviation and the maximum for job durations	20
Steps To Deadline	Average maximum number of jobs of a task that a single worker can finish until the deadline.	50
Task Concentration	The probability of new task submission for each time period.	0.35
Intended Schedule Density	Target assignment ratio for each time period.	0.2 - 0.7 (step 0.1)

Skill requirements are generated so that each skill with approximately equal probability either equals 0 which means that this skill is not required for the task, or is in $(0, 1]$ range. The random values for the $(0, 1]$ range are normally distributed (mean = 0.4, variance = 0.3).

The feedback that a consumer provides for a job is generated using the real skills of the worker which were assigned for this job. In contrast to the estimated skills, these real skills are unknown to the platform and are only used to simulate the real outcome quality (by calculating the suitability with these skills). This quality is thus reported as the feedback.

Crowd workers. The workers are assigned for jobs and return the result of job processing. Each worker has the claimed skills that s/he initially reports to the platform, and the real skills. The real skills are generated randomly with normal distribution with 0 mean and variance of 0.3. Then, the reported skills are initiated as real skills with injected error (normally distributed with mean value equal to the real skill and variance of 0.2). The crowd size in experiments was 1000 workers. This size is big enough to enclose the diversity of workers, but still allows for fast simulation. We tried to use 10000 instead, but the results did not change substantially. Workers can be unavailable at certain periods.

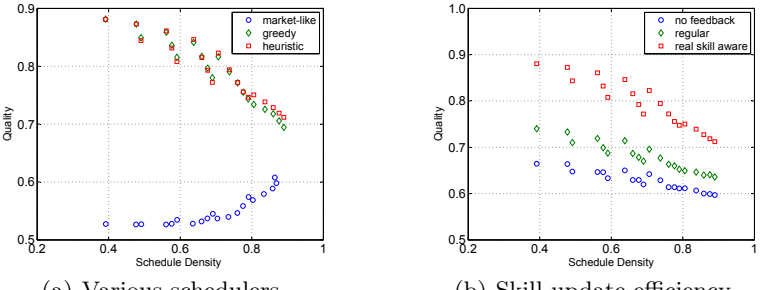
In our experiments we use a *Workers Unavailability* parameter which indicates the mean ratio of unavailable workers for each period of time (values used: $0.2 - 0.6$, step 0.1). The busy periods are generated randomly, but have a continuous form which reproduces human behavior. The amount of time that takes a worker to finish the job is the *Job Duration* with injected variations. In our experiments we used a value of 30%, which means that a job can be executed for $0.7 - 1.3$ of job duration. This reflects the random nature of the real world.

5.2 Experiment Types and Results

The aim of the experiments is to show the advantages of the platform's architecture. The first experiment type demonstrates the convenience of skill-based scheduling to the ordinary (random) task assignment. The second experiment type gives evidence of the skill update mechanism efficiency.

All the plots show the average job outcome quality for the resulting assignment density which is calculated as total number of periods for all workers while they were either unavailable or busy, divided by the difference between the first task submission time and the latest deadline. Apart from where explicitly specified, performed experiments contained no task deadline violations.

Various schedulers. To demonstrate the advantage of skill-based assignment, a scheduler which mimics a market-like platform was compared with the greedy scheduler and the heuristically-enhanced greedy scheduler (See Sect. 4). In market-like scheduling, the assignment followed the logic that randomly chosen workers were picking the most suitable for them active tasks. The results are shown in Fig. 3(a). In tests with high schedule density (about 0.8 or more), market-like assignment performed better than in tests with low density, because workers had more tasks to choose from. However, about 15% of task deadlines were violated in these tests, because workers aimed to fulfill their own preferences rather than the goals of the system. For the rest of the tests, the average quality was 1.5 times better for skill-based scheduling in the large. This clearly shows the benefit of skill-based scheduling. The heuristics did not improve the greedy algorithm substantially, and for some tests even impaired it.



(a) Various schedulers. (b) Skill update efficiency.

Fig. 3. Experimental results

Skill update efficiency. To demonstrate the efficiency of skill update mechanism, we compared the regular simulation which implements the logic described in Sect. 4 ("regular" series) to upper and lower bounds. The series named "no feedback" represents the lower bound and only the initial information on the profiles is used for scheduling. An upper bound to the algorithm is shown by the series of "real skill-aware". In this case the exact skills of a worker are known to the system. The improvement of the skill update mechanism over the lower bound is evident and keeps performing better at any scheduling density. In the experiments of Fig. 3(b) the improvement over no feedback remains between 10-15%. As Sect. 4.4 explains, the reason why it is never reaching real skill awareness is twofold. First, the scheduling strategy need some input right from start when only few feedback is available. Second, the feedback is a single value that describes the performance depending on ten different skills. Also, a skill value greater than required calculates the quality with the lowest value required. Even if there was enough data an accurate calculation would not be feasible in all cases. Thus, we decided to stick to a simpler quicker update algorithm that provides almost constant quality improvement and, after all, supports quality negotiation with a considerable and steady lower bound to make agreements.

The performance of scheduling and skill updating in a high workload test (10000 workers and 1000 tasks) was good enough for a period size of one minute. Thus, the performance is not a concern, since the real period size is likely to be bigger (e.g., 10 - 60 minutes).

6 Conclusion and Future Work

In this paper we proposed a skill-aware crowdsourcing platform model which allows to provide crowdsourcing services with SLAs and to control the task performance quality. In contrast to existing crowdsourcing platforms such as AMT, which follow a task market-oriented approach, our platform model is based on services computing concepts. Such a model is typically applied in enterprise workflow systems using, for example, the WS-HumanTask specification to design human interactions in service-oriented systems. However, WS-HumanTask and related specifications lack the notion of human SLAs and task quality. In our approach, negotiated SLAs and monitoring help to assign task requests to suitable workers. Thus, our platform ensures quality guarantees by selecting skilled workers. We introduced the proof-of-concept implementation with particular algorithms for task scheduling and worker profile management. The applicability of the platform design was proved in a simulated environment. The experimental results shows the clear advantage of skill-based scheduling in crowdsourcing, as the average quality is 50% better in the large comparing to the case when the workers choose tasks by themselves. The skill monitoring and updating mechanism improves the overall quality by 10-15%.

In our future work we will focus on workload and workforce availability predictions, SLA negotiation, and pricing in such scheduled crowdsourcing platforms. Also, we plan to extend the boundaries of our previous work [14] to consider

the scenarios where the SLAs between the crowdsoucing platform and a business process engine can be negotiated beforehand in an autonomic and adaptive fashion.

References

1. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: WSDM 2008, pp. 183–194. ACM (2008)
2. Agrawal, A., et al.: WS-BPEL Extension for People (BPEL4People) (2007)
3. Amazon Mechnical Turk (May 2011), `http://www.mturk.com`
4. Brabham, D.: Crowdsourcing as a model for problem solving: An introduction and cases. Convergence 14(1), 75 (2008)
5. Caprara, A., Monaci, M., Toth, P.: Models and algorithms for a staff scheduling problem. Math. Program. 98(1-3), 445–476 (2003)
6. Castellano, C., Fortunato, S., Loreto, V.: Statistical physics of social dynamics. Reviews of Modern Physics 81(2), 591–646 (2009)
7. Che, Y.: Design competition through multidimensional auctions. The RAND Journal of Economics 24(4), 668–680 (1993)
8. Doan, A., Ramakrishnan, R., Halevy, A.Y.: Crowdsourcing systems on the world-wide web. Commun. ACM 54, 86–96 (2011)
9. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153(1), 3–27 (2004), timetabling and Rostering
10. Ford, M., et al.: Web Services Human Task (WS-HumanTask), Version 1.0. (2007)
11. Howe, J.: The rise of crowdsourcing (June 2006), `http://www.wired.com/`
12. Ipeirotis, P.G.: Analyzing the Amazon Mechanical Turk Marketplace. SSRN eLibrary 17(2), 16–21 (2010)
13. Kern, R., Thies, H., Satzger, G.: Statistical Quality Control for Human-Based Electronic Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 243–257. Springer, Heidelberg (2010)
14. Khazankin, R., Schall, D., Dustdar, S.: Adaptive request prioritization in dynamic service-oriented systems. In: Proceedings of the 8th International Conference on Services Computing (to appear, 2011)
15. La Vecchia, G., Cisternino, A.: Collaborative Workforce, Business Process Crowdsourcing as an Alternative of BPO. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 425–430. Springer, Heidelberg (2010)
16. Leymann, F.: Workflow-Based Coordination and Cooperation in a Service World. In: Meersman, R., Tari, Z., et al. (eds.) OTM 2006. LNCS, vol. 4275, pp. 2–16. Springer, Heidelberg (2006)
17. LiveOps (May 2011), `https://www.livework.com/`
18. Parkes, D., Kalagnanam, J.: Models for iterative multiattribute procurement auctions. Management Science 51(3), 435–451 (2005)
19. Schall, D., Dustdar, S.: Dynamic Context-Sensitive Pagerank for Expertise Mining. In: Bolc, L., Makowski, M., Wierzbicki, A. (eds.) SocInfo 2010. LNCS, vol. 6430, pp. 160–175. Springer, Heidelberg (2010)
20. Sousa, J., Poladian, V., Garlan, D., Schmerl, B., Shaw, M.: Task-based adaptation for ubiquitous computing. IEEE Transactions on Systems, Man, and Cybernetics 36(3), 328–340 (2006)
21. Vukovic, M.: Crowdsourcing for Enterprises. In: Proceedings of the 2009 Congress on Services, pp. 686–692. IEEE (2009)
22. Yahoo! Answers (May 2011), `http://answers.yahoo.com/`