Skill Development Framework for Micro-Tasking

Shin Saito[1], Toshihiro Watanabe[2], Masatomo Kobayashi[1], and Hironobu Takagi[1]

[1] IBM Research – Tokyo, 5-6-52 Toyosu, Koto, Tokyo 135-8511, Japan
{shinsa,mstm,takagih}@jp.ibm.com
[2] The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo 113-8656, Japan
toshihiro_watanabe@mist.i.u-tokyo.ac.jp

Abstract. We propose a framework of micro-tasking that intrinsically supports the development of workers' skills. It aims to help developers of micro-tasking systems add skill development capabilities to their systems with minimal development costs. This will allow micro-tasking of skill-intensive work and improve the sustainability of micro-tasking systems. Based on the results of the micro-tasking projects we have carried out, our framework has three core modules: tutorial producer, task dispatcher, and feedback visualizer, which are supported by a back-end skill assessment engine. In closing, we discuss ways to apply the proposed framework to realistic micro-tasking situations.

Keywords: Crowdsourcing, Micro-Tasks, Skill Assessment, Skill Development, Gamification, Senior Workforce.

1 Introduction

Crowdsourcing provides an emerging type of labor market. There are two major types: macro-tasking and micro-tasking. Macro-tasks involve freelancers who work on complex tasks in projects or contests. Micro-tasks split a job into small pieces of work distributed to a number of workers via the Internet. Although both types of crowdsourcing call for skillful workers who can produce high-quality outcomes, particularly for micro-tasking there has been little attention to developing the skills of the workers, so the scope has generally been limited to lightweight tasks that need relatively less skill. The quality of the outcomes is controlled by filtering for workers who have higher skills, without considering the development of the individual workers' skills. However, if micro-tasking systems support skill development, they can produce outcomes of higher-quality and make their use more sustainable. This could expand the new labor market in two ways. First, it can provide younger workers with vocational training to learn job skills. Second, it can allow senior workers, who have advanced vocational knowledge but limited skills in information-communication technologies (ICT), to learn ICT skills so that they can do online work. The skill development support will make micro-tasking more suitable for more advanced tasks that call for expert skills.

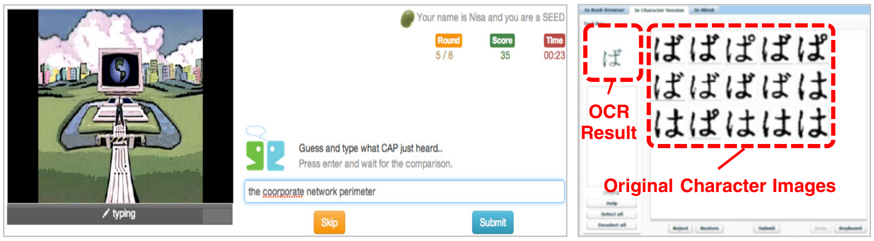
This paper proposes a micro-tasking framework that develops the skills of the workers. First we review the literature as well as two of our own micro-tasking projects, leading us to three capabilities the framework needs to have: tutorial producer, task dispatcher, and feedback visualizer. The tutorial producer generates learning materials based on task examples. The task dispatcher assigns a series of tasks to each worker based on the worker's learning curve. The feedback visualizer provides each worker with feedback about their efforts, their contributions, and the results of their skill development. The three functions require a back-end analytics module to analyze the results produced by each worker to assess the profiles of each worker, e.g., work accuracy, and the characteristics of each task, e.g., difficulty. Since the modules are encapsulated and connected with abstract interfaces, implementations of each module can be reused among different types of micro-tasking. This allows the developers to build micro-tasking systems that support skill development, by simply implementing task-specific operations.

This paper is organized as follows. In the next section, we review related work. Section 3 reviews our micro-tasking projects and discusses some implications. Section 4 describes the proposed framework. Then we discuss the applications of our framework in Section 5 and conclude the paper.

2 Related Work

Regarding worker education, macro-tasking services such as oDesk [29] and CrowdFlower [30] provide various training material and the learning history of each worker is used by requesters to choose workers. In contrast, skill development in micro-tasking receives a weaker focus. Amazon Mechanical Turk [31] and other micro-tasking platforms are interested in how to use inexpensive workers to produce more accurate results in less time. Although some studies have proposed frameworks to solve complex tasks via micro-tasking [1, 2], they paid little attention to skill development. One of the few exceptions is the work of Satzger et al. [3], which used the "tandem task assignment" approach to improve the skills of low-confidence workers. Kittur et al. [4] pointed out the potential of "crowd work-based education". Weld et al. [5] discussed personalized online education in crowdsourcing. They suggested maximizing learners' skills, whereas the typical goal in crowdsourcing is minimizing the costs to obtain high quality results. RABJ [6] tried to educate workers by selecting managers from the workers. These managers are responsible for creating task-specific guidelines and validating workers' output. Duolingo [32] allows users to learn foreign languages while contributing to crowdsourced translation work.

The assessment of workers' skills is essential for skill development. Several methods have been proposed with probabilistic models to simultaneously estimate both the skill of the workers and the difficulty of tasks [7-9]. There are also some studies that addressed task assignment based on the estimated skills [10, 11]. Tracking skill improvement over time is also essential. Donmez et al. [12] worked on the modeling of skills that are changing over time. However their system only handled short-term changes caused by fatigue. The changes were related to the time spent working, and were not affected by the properties of the tasks.


Your name is Nisa and you are a SEED

Guess and type what CAP just heard..
Press enter and wait for the comparison.

the coorporate network perimeter

(a) (b)

Fig. 1. Screenshots of (a) video captioning and (b) proofreading interfaces

3 Implications of Micro-Tasking Projects

We now describe two micro-tasking projects conducted by the authors and analyze them to clarify the requirements for our framework. In both cases the workers were unpaid volunteers and certain skills were required to complete the tasks. The two projects have similar implications for the skill development of the workers. These implications span many of the issues discussed in existing crowdsourcing research.

3.1 Crowdsourced Video Captioning

CapCap [13] provides video captions while also developing the linguistic skills of non-native workers (Fig. 1(a)). It is a gamified version of Collaborative Caption Editing System (CCES) [14], where each micro-task transcribes a short video segment (typically less than 10 seconds). A CapCap task is a game based on a similar concept to the ESP game [15], where workers iteratively caption the same segment, earning points in proportion to the degree of similarity between the outputs. CapCap has additional motivating features: levels, ranks, and teams. The level goes up as the worker's accumulated points increase. The system shows the team ranking and the level distribution within the worker's team. We introduced teams to motivate novices who could not earn high scores by themselves. CapCap was tested for three weeks in the authors' institution. See [13] for details of this experiment.

Findings. In the pilot period, 713 video segments with English narrations were distributed to 105 volunteer workers. A total of 60 segments were completed, with an average word error rate (WER) of 4.0%. The workers consisted of 15 native English speakers and 81 native Japanese speakers. The skill of each worker, which we defined as the average WER of the captions created by the worker, varied widely from 13.2% to 67.2% (among those who played more than 5 rounds). Due to the short pilot period, we could not verify its actual educational effect, though we will study this in the future. The workers generally rated the game design positively. However a novice who does not use English on a daily basis found "the content was too difficult for me", which suggests that the skill gaps could demotivate workers. Some novice workers criticized the motivational system, saying that it was too hard to get to higher levels. Such comments indicate that slightly different incentives are needed for skill-focused workers versus task-oriented workers versus the most skilled workers who produced the highest quality results. We want to maximize overall participation by recognizing and responding to the motivational differences among these three groups of workers. Other comments included "I did not know how to play the game" and "I wanted to know the scoring rules", even though CapCap included instructions. Since it was a real-time game, the simple static instructions were inadequate.

3.2 Crowdsourced Proofreading

EBIS (E-Book Improvement System) [16, 17] crowdsources proofreading, with the workers correcting OCR (Optical Character Recognition) errors (Fig. 1(b)). The system has multiple types of proofreading tasks and an online forum where workers can communicate with each other and ask questions about the tasks. The system allows workers to work on their preferred types of tasks. Each worker received visual feedback about their task history, such as the number of contributed books and the worker ranking based on the amount of completed work. Prior to the project launch, we held a full-day introductory session for the initial workers. We also provided them with printed manuals. The project started in October 2013, with an open participation policy. We analyzed the data for approximately the first three months of operations. See [18] for details of the experiment.

Findings. As of January 2014, 178 workers had registered, with 112 who did at least one task. They contributed more than 1,200 hours of work and completed 182 books. The attendees of the introductory session liked the session and the manuals, confirming the findings in [19, 20]. Those instructions were designed mainly for older workers who had no prior experience with crowdsourcing. However they should also be useful for other workers, since the micro-tasks in EBIS involve a number of editorial rules that workers had to learn. A total of 60 questions about the editorial rules and other topics were posted in the forum. Some of them had been addressed in manuals, but others were undocumented. The already addressed questions were asked mostly due to difficulties in understanding the rules, while the new topics were mostly related to the difficulty of the tasks. The preferences for task types varied among the workers. With regard to the visual feedback, Itoko et al. [18] reported that the visualization of their contributions was preferred by the workers of all ages, while the younger workers liked the ranking scores more than the seniors did. A worker commented that she strongly wanted feedback on the accuracy of her work rather than rankings, which indicates the potential for personalized feedback.

3.3 Implications

Here are three similarities among the two projects: *importance of instructions, need for appropriate task assignment,* and *effectiveness of feedback to workers*. These findings lead to the following requirements for a skill development framework that allows gamified learning: (1) support for tutorial generation, (2) task assignment for gradual learning, (3) motivating feedback by visualization, and (4) back-end skill assessment engine that provides these three components with estimated skill information by analytics.

Support for Tutorial Generation. Although interactive lectures and printed instructions are effective, it is expensive to prepare them. While automatic manual generation techniques like GraphScript [21] may reduce the costs, human work is still needed. One alternative approach is RABJ [6], which selects *managers* from the workers, and these managers create guidelines for the tasks. As an alternative to interactive lectures, online interactive tutorials [22] may work well. Many modern computer games have interactive tutorials using the same user interface as the game itself. Such tutorials are provided in chunks of gradually increasing complexity and users naturally learn how to play the game [23]. Since authoring such tutorials is costly work, systems to help create tutorials from example tasks and from workers' behaviors are needed to reduce the costs.

Task Assignment for Gradual Learning. In our experiment with CapCap, the range of skills among workers varied widely, ranging from native English speakers to non-native beginners. The task difficulty also varied from studio-recorded content to a conversation among four people in a noisy environment. As pointed out by the novice worker in the experiment, the difficulty of the tasks assigned to a worker should be suitable for the worker's skills. After a task is completed, the worker may try a slightly more difficult task. This idea matches the evolving-skill model discussed in [3] as well as the *gamenics* theory [23]. Gamenics is a design principle for video games, which has two objectives: intuitive operability and gradual learning.

Motivating Feedback by Visualization. The visual feedback is effective motivation for workers (e.g., [24]). Motivation is divided into *intrinsic* and *extrinsic* motivation [25]. For example, progress visualization and content personalization improve intrinsic motivation, while social mechanisms such as competition work for extrinsic motivation. In the EBIS project, which aimed to benefit from both types of motivation, the young workers tended to prefer feedback for extrinsic motivation while the seniors had the opposite preference. Since there are a number of design choices for visualization, it is desirable that the administrator of the system be able to easily choose and change those settings.

Skill Assessment. Accurate assessment of workers' skills plays a crucial role in all of these three requirements. The estimated skills can be used to select appropriate tutorial content and micro-tasks for each worker as well as to present feedback for their progress in skill development. Various skill models can be used. For example, if we use the multi-dimensional skill model of Welinder et al. [8] for CapCap, the skills will

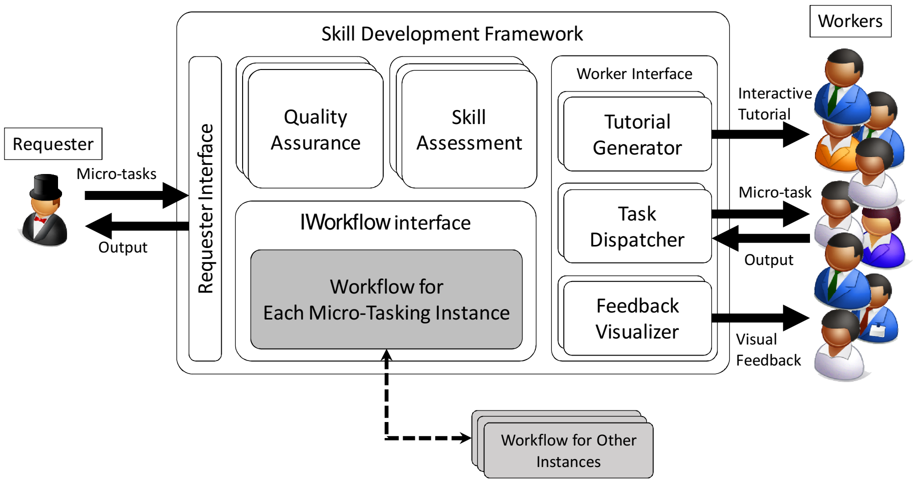


Fig. 2. Component diagram of the proposed framework

be represented as a combination of listening, writing, and keyboard typing skills. In EBIS, the skills consist of a sub-skill for each task type and each sub-skill is further divided into latent skills (ICT skills, visual acuity, etc.). Since a number of skill models have been proposed (see Related Work) and it is difficult to determine the best model in advance, the framework should allow easily switching skill models to find the most appropriate model through a tailoring approach.

4 Framework

Based on the discussion in the Section 3, we propose a micro-tasking framework that takes into account the development of each worker's skill, which is greatly different from the earlier micro-tasking frameworks.

4.1 Overall Architecture

The architecture of the proposed framework (Fig. 2) is inspired by OSGi [33], a Java modularity framework based on a plug-in architecture, where loosely-coupled components communicate with each other via standard or instance-specific APIs. It consists of core components including Tutorial Producer, Task Dispatcher, Feedback Visualizer, and Skill Assessment Engine, which correspond to the four functions discussed. It also has a Quality Assurance Engine, which tries to produce the best aggregate result for each requester considering the assessed skills, as well as shared libraries that help developers reuse frequently used functions among each instance. For example, the skill estimation algorithms used in the Skill Assessment Engine, the output merging algorithms used in the Quality Assurance Engine, and some user interface components, including the logging-in/out features, are shared. The workflow defines the instance-specific data and control flows, as described in Section 4.2.

4.2 Workflow

The developer of a micro-tasking instance needs to implement its own workflow class that implements an *IWorkflow* interface for each instance. Workflow class implements callback methods which will be called from the system runtime with an execution context. For example, the framework handles the log-in process and after logging-in, the callback will be invoked to show the top page for each worker.

Here are the typical data and control flows. First, the micro-tasks are submitted by a task requester with requirements, which can include the desired speed, cost, or quality. Then their difficulty (and possibly the skill of the workers) is estimated by the Skill Assessment Engine. When a worker wants to do a task, the Task Dispatcher assigns the most relevant task and the worker creates the *output* for the task. A task can be assigned to more than one worker to generate higher confidence results. If enough output is collected, all of the outputs for that task is collected by the Quality Assurance Engine and merged using the estimated quality. The result is returned to the requester with such properties as the confidence of estimation.

4.3 Core Components

Here we consider CapCap and skill assessment by Welinder et al. [8] as examples, where the workers are denoted by $u_i \in U$ and each CapCap task, i.e., a video segment, is denoted by $t_j \in T$, whose skill is expressed as a *d*-dimensional vector, $\boldsymbol{w}_i \in [0,1]^d$.

Skill Assessment Engine. Skill Assessment Engine plays a crucial role in the framework. We define a *work* element as a triplet of a user, a task, and its output, denoted by $(u_i, t_j, l_{ij}) \in U \times T \times L = W$, where l_{ij} is an output for task t_j by a worker u_i. This engine maintains a work history, which stores all of the work information, using $H = \mathfrak{P}(W)$. Then the engine computes or estimates the current skill of each worker, based on the history. It can also estimate the skill improvements by dividing H into time segments. These estimates are used by the other core components to make decisions on the workflow.

Tutorial Producer. This component provides helper functions for generating the graded-difficulty learning tutorial. For example, it provides sample task lists ordered by estimated difficulties, or it can identify the difficult tasks for which many workers produced wrong answers with the support of the Skill Assessment Engine. The developer needs to provide a tutorial user interface. Typically this is almost the same as the actual task UI, but runs when the *tutorial* flag is set to true.

Task Dispatcher. This component dispatches the most relevant micro-task to a specified worker with the help of the Skill Assessment Engine. Based on how the skills of each worker improve over time, it assigns the appropriate task to the worker. For example, if a worker is a novice, then easier tasks are assigned, while if the system (or the requester) wants a worker with intermediate skills to improve, then it dispatches challenging tasks to that worker.

Feedback Visualizer. This component provides visual feedback to motivate workers. Visualization includes showing the worker's skill set that changes over time, a work history including accuracy, and recommended tasks, as well as social-network-related visualizations such as competitive or collaborative features, with the help of the Skill Assessment Engine.

5 Discussion

5.1 Implementation Patterns Using the Framework

We describe several implementation patterns with this framework using EBIS and CapCap as examples. First, implementing EBIS from scratch using this framework would be straightforward. The developers would only need to implement EBIS-specific features such as task decomposition, which breaks down a single book into micro-tasks, output merging, and the user interface. They can benefit from the shared libraries including the worker account management, task assignment, and skill management. Starting from an EBIS implementation, CapCap could be easily implemented using this same framework. The developers would need to implement the CapCap task decomposition, the other components including the output merging and user interface can be reused from shared library or from EBIS. As a third case, if EBIS has already been implemented without using the framework, then the framework could be used to enhance EBIS to support skill development, but additional work would be needed. The current implementation would need to be refactored to match the framework APIs. In the first and second cases, the developments cost would be greatly reduced by using the framework. In the third case, though the costs would be higher than in the first and second cases, the code would be better and future modifications of the system would be much easier.

5.2 Applicability of the Framework

We believe the proposed framework will be applicable to a wide variety of micro-tasking applications. Our framework easily incorporates micro-tasking frameworks that focus on the quality of the output. For example, Dai et al. [26] and Lin et al. [9] use decision theoretic algorithms to assign tasks, where they compute a utility U to decide whether to dispatch a task to one of the workers or to generate final result. We can define and add new utility by considering the skill development of the workers, for example, by defining a utility based on improvement of skill when a task t_j would be assigned to a worker u_i.

We introduced the visualization as a motivational mechanism. What about other motivating mechanisms or incentives? It has been reported that extrinsically motivating factors like monetary incentives work for improved speed, but do not improve the quality, while intrinsic motivations improve accuracy [27, 28]. From the perspective of social learning, the framework could incorporate techniques that Weld et al. introduced [5] for more effective learning by the workers.

6 Conclusion

This paper proposed a micro-tasking framework that supports skill development of individual workers. We designed the framework based on the review of two micro-tasking projects we have conducted. It consists of four key components, a tutorial generator, a task dispatcher, a feedback visualizer, and a skill assessment engine. The framework will help developers create sustainable micro-tasking systems. Also, it will expand the scope of micro-tasking, by supporting new types of tasks, such as skill-intensive work, and new types of workers, such as senior citizens. Our future work will includes more investigations of practical problems that must be addressed when using the proposed framework in realistic systems. We hope this will leads to the development of a reference implementation.

Acknowledgements. We appreciate the help of Kaoru Shinkawa in analyzing the data from CapCap. This research was partially supported by the Japan Science and Technology Agency (JST) under the Strategic Promotion of Innovative Research and Development Program.

References

1. Kittur, A., Smus, B., Khamkar, S., Kraut, R.E.: Crowdforge: Crowdsourcing complex work. In: Proc. UIST 2011, pp. 43–52. ACM (2011)
2. Kulkarni, A., Can, M., Hartmann, B.: Collaboratively crowdsourcing workflows with turkomatic. In: Proc. CSCW 2012, pp. 1003–1012. ACM (2012)
3. Satzger, B., Psaier, H., Schall, D., Dustdar, S.: Auction-based crowdsourcing supporting skill management. Information Systems 38(4), 547–560 (2013)
4. Kittur, A., Nickerson, J.V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M., Horton, J.: The future of crowd work. In: Proc. CSCW 2013, pp. 1301–1318. ACM (2013)
5. Weld, D.S., Adar, E., Chilton, L., Hoffmann, R., Horvitz, E., Koch, M., Landay, J., Lin, C.H., Mausam, M.: Personalized online educationa crowdsourcing challenge. In: Proc. HCOMP 2012 (2012)
6. Kochhar, S., Mazzocchi, S., Paritosh, P.: The anatomy of a large-scale human computation engine. In: Proc. HCOMP 2010, pp. 10–17. ACM (2010)
7. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.R.: Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: Proc. NIPS 2009, pp. 2035–2043 (2009)
8. Welinder, P., Branson, S., Belongie, S., Perona, P.: The multidimensional wisdom of crowds. In: Proc. NIPS 2010, pp. 2424–2432 (2010)
9. Lin, C.H., Mausam, M., Weld, D.S.: Crowdsourcing control: Moving beyond multiple choice. In; Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence (2012)
10. Yan, Y., Fung, G.M., Rosales, R., Dy, J.G.: Active learning from crowds. In: Proc. ICML 2011, pp. 1161–1168 (2011)
11. Wauthier, F.L., Jordan, M.I.: Bayesian bias mitigation for crowdsourcing. In: Proc. NIPS 2011, pp. 1800–1808 (2011)

12. Donmez, P., Carbonell, J., Schneider, J.: A probabilistic framework to learn from multiple annotators with time-varying accuracy. In: Proc. SD 2010, pp. 826–837 (2010)
13. Kacorri, H., Shinkawa, K., Saito, S.: Introducing game elements in crowdsourced video captioning by non-experts. In: Proc. W4A 20114. ACM (2014) (in press)
14. Sobhi, A., Nagatsuma, R., Saitoh, T.: Collaborative caption editing system—enhancing the quality of caption editing system. In: CSUN 2012 (2012)
15. von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proc. CHI 2004, pp. 319–326. ACM (2004)
16. Ishihara, T., Itoko, T., Sato, D., Tzadok, A., Takagi, H.: Transforming Japanese archives into accessible digital books. In: Proc. JCDL 2012, pp. 91–100. ACM (2012)
17. Kobayashi, M., Ishihara, T., Itoko, T., Takagi, H., Asakawa, C.: Age-based Task Specialization for Crowdsourced Proofreading. In: Stephanidis, C., Antona, M. (eds.) UAHCI 2013, Part II. LNCS, vol. 8010, pp. 104–112. Springer, Heidelberg (2013)
18. Itoko, T., Arita, S., Kobayashi, M., Takagi, H.: Involving senior workers in crowdsourced proofreading. In: Stephanidis, C., Antona, M. (eds.) UAHCI/HCII 2014, Part III. LNCS, vol. 8515, pp. 106–117. Springer, Heidelberg (2014)
19. Leung, R., Tang, C., Haddad, S., McGrenere, J., Graf, P., Ingriany, V.: How Older Adults Learn to Use Mobile Devices: Survey and Field Investigations. ACM Trans. Access. Comput. 4(3), Article 11 (2012)
20. Kobayashi, M., Ishihara, T., Kosugi, A., Takagi, H., Asakawa, C.: Question-Answer Cards for an Inclusive Micro-Tasking Framework for the Elderly. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013, Part III. LNCS, vol. 8119, pp. 590–607. Springer, Heidelberg (2013)
21. Huang, J., Twidale, M.B.: Graphstract: Minimal graphical help for computers. In: Proc. UIST 200707, pp. 203–212. ACM (2007)
22. Rowe, S.C., Cohen, C.J., Tang, K.K.: Applying computer game tutorial design techniques to simulation-based training. In: Proc. 2004 Fall SIW (2004)
23. Isbister, K., Schaffer, N.: “gamenics” and its potential. In: Game Usability: Advancing the Player Experience. CRC Press (2008)
24. DiMicco, J., Millen, D.R., Geyer, W., Dugan, C., Brownholtz, B., Muller, M.: Motivations for social networking at work. In: Proc. CSCW 2008, pp. 711–720. ACM (2008)
25. Kaufmann, N., Schulze, T., Veit, D.: More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk. In: Proc. AMCIS 2011 (2011)
26. Dai, P., Mausam, W.D.S.: Decision-theoretic control of crowd-sourced workflows. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
27. Mason, W., Watts, D.J.: Financial incentives and the “performance of crowds”. SIGKDD Explor. Newsl. 11(2), 100–108 (2010)
28. Rogstadius, J., Kostakos, V., Kittur, A., Smus, B., Laredo, J., Vukovic, M.: An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. In: Proc. ICWSM 2011 (2011)
29. oDesk, http://www.odesk.com
30. CrowdFlower, http://crowdflower.com
31. Amazon Mechanical Turk, http://www.mturk.com
32. Duolingo, http://www.duolingo.com
33. OSGi, http://www.osgi.org/Specifications