ReTool: Interactive Microtask and Workflow Design through Demonstration

Chen Chen[1] **Xiaojun Meng**[1] **Shengdong Zhao**[1] **Morten Fjeld**[2]

[1]NUS-HCI Lab, National University of Singapore, Singapore
{chenc, xiaojun, zhaosd}@comp.nus.edu.sg

[2]t2i Lab, Chalmers University of Technology, Gothenburg, Sweden
fjeld@chalmers.se

ABSTRACT

In addition to simple form filling, there is an increasing need for crowdsourcing workers to perform freeform interactions directly on content in microtask crowdsourcing (e.g. proofreading articles or specifying object boundary in an image). Such microtasks are often organized within well-designed workflows to optimize task quality and workload distribution. However, designing and implementing the interface and workflow for such microtasks is challenging because it typically requires programming knowledge and tedious manual effort. We present ReTool, a web-based tool for requesters to design and publish interactive microtasks and workflows by demonstrating the microtasks for text and image content. We evaluated ReTool against a task-design tool from a popular crowdsourcing platform and showed the advantages of ReTool over the existing approach.

Author Keywords

ReTool; Crowdsourcing; Workflow; Freeform Interactive Microtasks; Programming by Demonstration

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous.

INTRODUCTION

With the emergence of platforms such as Amazon Mechanical Turk (AMT) [1] and CrowdFlower (CF) [9], microtask crowdsourcing has emerged as a popular research topic in HCI. In the microtask crowdsourcing process, a problem is divided into a set of small tasks (microtasks) which can be completed independently by workers in a short period of time for monetary rewards. For each microtask, the requester often needs to provide a piece of content as input (e.g. an audio file) and asks the worker to process the content to generate the output (e.g. text transcript of the audio file). Sometimes, the output of one microtask serves as the input for another microtask. Such an interdependent arrangement of microtasks is called the workflow, which is often used to break down a complex task into multiple steps to improve the overall quality of the results, or to optimize the workload distribution among workers.

Many microtasks (e.g. image tagging, media transcription, etc.) only require workers to answer questions regarding the content by filling out simple forms (e.g. text field, checkbox, radio buttons) without directly interacting with the content. Such form-based microtasks are relatively straightforward to design and deploy: both AMT and CF platforms provide tools to help requesters construct such forms from scratch or modify them from existing templates.

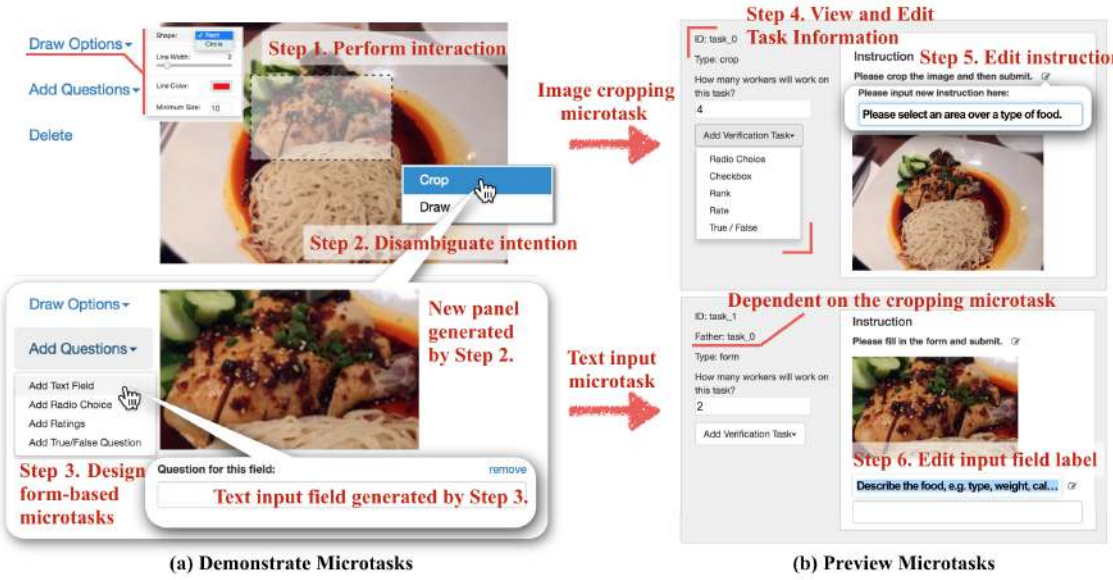
However, not all microtasks can be completed using form filling alone; recently, there has been an increasing number of microtasks that require freeform interactions on the content (e.g. performing a set of text selection and modification interactions to proofread an article [5]; dragging content to form clusters [3]; drawing bounding boxes over specific objects in an image [21, 12, 20]; or marking specific time points on a video clip [17, 13]).

In contrast to form-based microtasks, the interface and interactions design of interactive microtasks is content dependent (e.g. text content requires a different interface and set of interactions from an image or a video), and is more complex to implement. Existing microtask platforms such as AMT and CF do not provide direct support for the design of interactive microtasks. Researcher-developed tools such as CrowdWeaver [14], CrowdForge [15] and Turkomatic [16] can support the design of workflows; but they also lack support for designing interactive microtasks. The only alternative is to use programming-based approaches, such as Turkit [19] and AMT SDKs [2]; however, the need of programming skills sets a significant threshold for many requesters.

To lower the barrier of entry for designing and deploying interactive microtasks with workflows, we developed ReTool, a web-based tool that simplifies the design and deploy of interactive microtasks with workflows by applying the "Programming by Demonstration" (PbD) concept [11]. The original PbD defines a mechanism whereby the user writes a program by giving an example of what the program should do. In our context, the PbD refers to the mechanism by which the requester designs interactive microtasks with workflows by giving an example of how the tasks can be completed. Working with ReTool, the requester first uploads a piece of sample content. Depending on the content type, a content specific workspace is created. The requester then performs a sequence of interactions (e.g. tapping-and-dragging, clicking) on that sample content within the workspace. The performed inter-

CHI 2017, May 06 - 11, 2017, Denver, CO, USA

ACM 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025969



(a) Demonstrate Microtasks (b) Preview Microtasks

Figure 1. The flow of ReTool to create interactive microtasks with a workflow.

actions are recorded and analyzed to generate the interactive microtasks with workflows. As a proof of concept, ReTool first provides support to design text and image based interactive microtasks with workflows, but the approach can easily be extended to other types of content such as audio and video.

The contributions of this paper are as follows:

1. A new crowdsourcing microtask design approach which adopts the concept of "Programming by Demonstration".
2. A functional prototype of the proposed approach, where "Programming by Demonstration" can happen.
3. An empirical experiment that demonstrates the capability and usability of the proposed approach in creating interactive microtasks with workflows.

USAGE SCENARIO

John photographs his family's meals as a way to log their life. He has a great deal of dietary imagery and now he wants to know whether his family members are eating healthily. Since the workload of identifying each food item in each image is quite high, John decides to seek help from crowdsourcing. He wants the crowd to describe each individual food item in an image instead of describing the image as a whole. Thus he requires workers to firstly crop out each food item in an image and then describe the cropped item.

John creates a new project, and follows ReTool's instructions to upload a sample of content (a dietary image) to the project. ReTool generates an image specific workspace for John's project as shown at the top of Figure 1(a). The interface of the workspace has two main components. The left side is an option panel with "Draw Options" to customize the annotations drawn on images (such as annotation shape, stroke size and color), "Add Questions" to add common form elements (such as multiple choice and text field) to support the design of form-based microtasks, and "Delete" to delete the content. The right side is the content panel containing the uploaded image, where John can demonstrate microtasks.

To demonstrate the microtasks for his project, John firstly selects a region covering a food item on the image as step 1 in Figure 1. Upon the release of the mouse button, ReTool brings up a popup menu to ask John whether he wants to crop the image or to draw a rectangle as step 2 in Figure 1. John chooses the "Crop" option, and sees a new panel generated containing the selected area of the image. He selects the "Add Text Field" from the bottom option set to add a text input field for the new panel, which will be placed under the cropped image as step 3 in Figure 1.

Once he clicks "Next", ReTool processes John's interactions and generates a preview of the interface of two microtasks as shown in Figure 1(b). One microtasks asks workers to select an area to be cropped from the image, and the other asks workers to input a description of the cropped area. On the preview interface, John can define the number of workers associated with each microtask, add verification tasks, set the instruction for each microtask and modify the question for the input field as steps 4, 5 and 6 in Figure 1. Based on his input as shown in Figure 1(b), ReTool generates two interdependent microtasks for each image: two workers are going to input descriptions for each of the four cropping results of the original image. After John is satisfied with the microtask and workflow, he proceeds to the next step and ReTool directs him to upload all the dietary images to the project. Finally, a URL is generated for John to deploy the task online.

As illustrated in this scenario, requesters can perform the following operations in the task demonstration workspace. *(1) Perform interactions.* A requester demonstrates the interaction (e.g. clicking, selection, or highlighting) on the content panel. *(2) Disambiguate intention.* As the same interaction may have different intentions, (e.g. selecting a region of an image could mean to either crop or annotate), a popup menu

Supported content types and interactions	
Text	Editing (modification), selection (extraction)
Image	Selection (annotation, cropping)
Supported form elements	
text field, multiple choice, ratings, true/false question	

Table 1. Content types, interactions and form elements currently supported by ReTool.

with the possible intentions will appear after an interaction to seek clarification. *(3) Set properties.* Sometimes the requester may have a specific requirement on the appearance or format for the annotation added to the content. To satisfy this need, a set of "Draw Opitons" is provided to allow adjustments. *(4) Add form-based elements.* ReTool also provides support for designing *form-based microtasks*, where the requester can simply insert form elements from the option list. The interactions and form elements ReTool currently supports are shown in Table 1.

RETOOL IMPLEMENTATION

ReTool is developed to work as a crowdsourcing platform including: (1) creating crowdsourcing projects; (2) designing microtasks and workflows; (3) previewing and adding verification tasks; and (4) publishing tasks. The first step is straightforward, so we will focus on describing the implementation of steps 2, 3 and 4 below.

Microtask and Workflow Generation

We identified the four basic types of crowdsourcing workflows as shown in Figure 2: (1) Parallel - multiple workers work on microtask A [6, 22, 10, 17]; (2) Sequential - microtask B depends on results of microtask A [5, 13, 12, 8]; (3) Conditional - if a condition is true do microtask A, otherwise do microtask B [7]; (4) Looping - do microtask A until a condition is true [4]. The four workflows can be nested to form complex workflows (e.g. [21] uses nested conditional and looping workflow; [20] and [3] use nested sequential and looping workflow). Parallel workflow is the simplest type since the tasks involved have no dependency on one another. Sequential workflow is described below. Conditional and looping workflows are described in the next subsection.

Supporting workflow design involves a key challenge of how to conceal the complexity of workflow design from requesters, as we want them to focus on demonstrating tasks instead of thinking about workflow design. This means ReTool needs to be able to appropriately translate user interactions into one or more microtasks with their workflows. While generating sequential workflow, ReTool makes two observations: (1) whether an interaction generates a new piece of content and (2) whether the requester performs a subsequent interaction on the newly generated content. If both occur, a sequential workflow is designed. For example, a cropping interaction on an image or an extraction from a piece of text will result in a new image/text segment. Performing interactions on the new image/text segment or adding some forms to the new segment will result in a new microtask which depends on the previous cropping or extraction microtask. Thus, sequential workflow is generated and the newly generated microtask will be regarded as a descendant of the previous one.

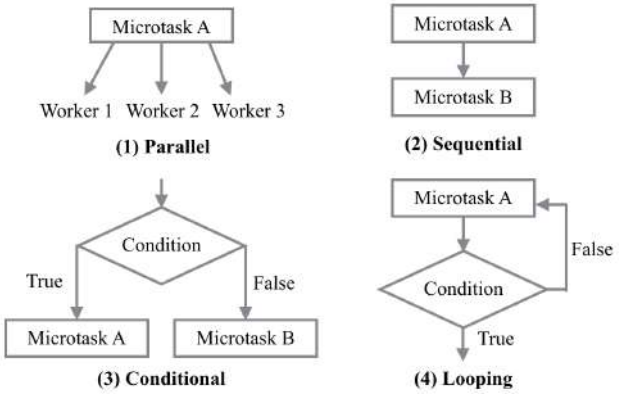


Figure 2. The four basic types of crowdsourcing workflows.

To generate the final workflow, ReTool also deals with two more conditions: (1) If the requester erroneously performs an interaction which generates a new piece of content, s/he can delete the new content by clicking the "Delete" button in the option panel. With this operation, all other descendant content will also be deleted, and the workflows related to the deleted content will not be generated. (2) The requester may repeatedly perform the same interaction and intention on a piece of content (e.g. annotating or cropping multiple regions). In this case, such interactions will be merged and generate only one microtask for multiple workers to work on.

Previewing Microtask Interface & Workflow

In this step, the requester can preview the interface and workflow of microtasks generated in the second step, and edit the instructions for the microtasks and other properties (e.g. worker number). The requester can also add verification tasks and add an advanced workflow at this step.

Verification Task Generation. Verification tasks can be inserted after a normal microtask to verify its results. A verification task selects the best result or checks the correctness of a result but will not change the result's content, thus will not break the original workflow. ReTool currently supports five basic types of verification tasks: radio button choice, checkbox, rating, ranking, and the true/false question.

Conditional and Looping Workflows Generation. True/false verification can also be used to create advanced workflows: conditional and looping workflows. The true/false question can be regarded as a condition controller. In configuring a true/false verification task, the requester can decide whether s/he wants an ancestral microtask of the verification task to be re-performed if the "false" condition is chosen. For example, in the usage scenario described previously, John can add a true/false question to verify whether the cropped part is a valid object and add another true/false question to verify the tags; then choose to recreate the cropping or tagging microtasks respectively if the verified result is false.

Publishing Microtasks

After previewing, the requester can upload the collection of content to be crowdsourced. Some microtasks are created immediately; others will be created after their parent tasks are

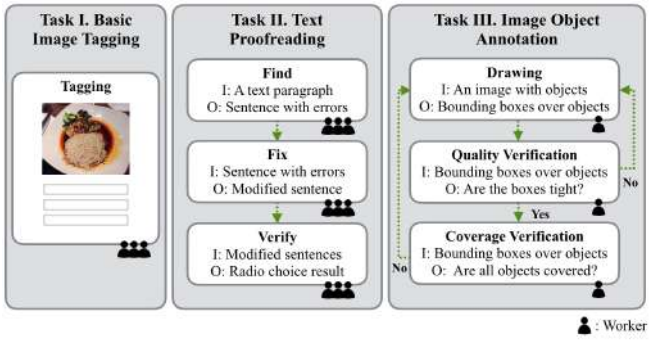


Figure 3. The three types of tasks to be designed (I: input; O: output).

completed by one or more workers. All uncompleted microtasks of the same project are maintained in a FIFO (first in, first out) queue and a common URL will be generated to access the queue. A worker will be assigned with an available microtask through the URL from crowdsourcing marketplaces or social network platforms. The results of the worker will be recorded in ReTool. By now, ReTool can support the entire process from designing, publishing, to gathering results.

EVALUATION

ReTool is designed to lower the entry barrier for requesters to design and deploy interactive microtasks with workflows. We are interested to know: (1) Can requesters without programming knowledge use ReTool to complete interactive microtasks with workflows design? (2) Does ReTool make the job easier for requesters with programming knowledge? With these goals in mind, we conducted a user study to find out how potential requesters with varying programming skills use ReTool to design a number of different interactive microtasks with workflows. To get a sense of how it compares with existing approaches, we also included a lower bound baseline, Amazon Mechanical Turk, as an alternative approach in the study.

Participants. 14 participants (7 females, 7 males) ranging from 21 to 25 years old (M=23.3, SD=1.35) were recruited from the university community. 6 participants were from engineering or arts & social science and self-reported as having little or very basic programming knowledge. The other 8 were either 3rd or 4th year computer science students and self-reported as knowledgeable programmers. In the pre-questionnaire, 6 participants (all from the knowledgeable programmers) reported that they were somewhat familiar with crowdsourcing (i.e. they knew the concept of crowdsourcing, but had not tried to publish a task). The other 8 reported they knew only the name or had never heard about it before.

Procedure and Tasks. We began by giving the participants a 20-minute explanation of what crowdsourcing is and what crowdsourcing microtasks and workflows look like, and answered their questions. In this way we attempted to give all participants a rudimentary knowledge of crowdsourcing and task design. In the next 20 minutes, we used two crowdsourcing tasks to teach the participants how to design tasks, workflows and verification tasks using ReTool; how to use templates to design tasks; how to write HTML/JS code to customize the tasks; and how to design multiple independent tasks to support workflows using AMT online task design tool.

After a 10-minute break, the participants were asked to use both ReTool and AMT to design microtasks and workflows for three crowdsourcing tasks: (I) basic image tagging (workers give three tags to describe each image); (II) text proofreading (workers proofread text content following the *Find-Fix-Verify* protocol described in [5]), and (III) image object annotation (workers detect a particular object in each image by drawing tight bounding boxes around every instance of the object, following the *Drawing-Quality verification-Coverage verification* protocol described in [21]). The three tasks were selected for their varying complexities as shown in Figure 3: task I was an independent form-based microtask; task II included two text based interactive microtasks and a verification task using sequential workflow; task III included one image based interactive microtask and two verification tasks using a nested conditional and looping workflow. In combination, these tasks covered two input styles (form-based and interactive), two content types (text and image), and all four types of basic workflows (used in isolation or in combination). To avoid order effects, the order of approaches (ReTool vs. AMT) was counter balanced.

We used two criteria to decide whether the participants had succeeded in a task: (1) the design was finished within time constraints; (2) the microtasks accessed by the URL were as described. For example, a participant succeeded in task II only if s/he designed the three microtasks and workflow as shown in Figure 3 within the given amount of time (10 minutes).

Results. For the AMT condition, 13/14 participants succeeded in task I, taking 6.2 minutes on average. However, no participant (even those who were proficient programmers) finished either task II or III within the time constraint.

For the ReTool condition, the same number of participants (13/14) succeeded in task I, taking 4.6 minutes on average. In addition, 9/14 participants succeeded in task II, taking 7.5 minutes on average; and 10/14 participants succeeded in task III, taking 5.7 minutes on average. Among the 9 and 10 participants who finished task II and task III, we were encouraged to find that 4/9 for task II and 2/10 for task III were non-programmers. This showed that 67% (4/6) and 33% (2/6) of non-programmers could succeed in tasks II and III. The lower number for success in task III may be due to higher difficulty for non-programmers to understand looping workflow than sequential workflow. We also found that 75% (6/8) and 50% (4/8) of participants who originally knew little about crowdsourcing and task design could succeed in tasks II and III. The results show that ReTool is able to help non-programmers and new crowdsourcers to design complex microtasks and workflows in a fairly short time.

Among the 5 participants who failed task II, 3 completed the *Find* and *Fix* microtasks, but forgot to add the verification task. Only 2 ran out of time. Of the 4 participants who failed

task III, all 4 completed the task within the time constraint, but 2 forgot to add the third verification task, and the other 2 erroneously selected the ancestral microtask to be reperformed in the third yes/no verification task.

At the end of the experiment, the participants completed a post-questionnaire adapted from the Computer System Usability Questionnaire [18], with 12 closed-ended questions using a 7-level Likert scale (1=strongly disagree, 7=strongly agree). The scores for ReTool ranged from 4.2 to 5.8 (M=5.14), which is significantly higher ($p<.05$) than scores for AMT, from 1.9 to 3.1 (M=2.34). Participants reported for ReTool that "*it is easy to design verification tasks*" (5.8), "*it is easy to split tasks*" (5.64) and "*it is easy to use*" (5.64). P5 wrote "*(ReTool is) very intuitive and easy to use, (it) gives users a smooth process of setting up a project*". However, as a proof of concept, ReTool certainly does not provide all the functionalities and can still be improved in its intuitiveness. This is reflected in the lower scores for questions regarding "*it does everything I expect it to do*" (4.43) and "*I can use it without written instructions*" (4.2). P1 mentioned that "*more interactions can be added*".

DISCUSSION AND CONCLUSION

Combining the quantitative and qualitative results, we conclude that by leveraging PbD, ReTool can successfully lower the barriers for requesters to design interactive microtasks with workflows. Further, ReTool allows programmers to design interactive microtasks with workflows in a much shorter time than when using alternative systems, and ReTool enables non-programmers to design interactive microtasks with workflows that they would not be able to design otherwise due to lack of programming skills.

The PbD approach is well suited for designing tasks which require collecting information through interactions and manipulating content. However, the PbD approach would fail if a required interaction of the task is unsupported (e.g. take a photo using smartphone), or if involving computational tasks that are hard to interactively demonstrate. The latter is possible if a requester wants to design tasks that involve both human and machine intelligence, where the computational aspect of that task may not be designable through human demonstration.

As a proof-of-concept prototype, ReTool also has limitations and can be extended as future work. For example, (1) supporting interactive microtasks that involve video and audio content; (2) incorporating qualification tasks before accepting a worker; (3) interacting with multiple pieces of content in one microtask (e.g. when a microtask has multiple images, or mixed images and text); (4) supporting clustering interactions as in crowd synthesis [3]. Most of these features can be supported with additional engineering work.

In conclusion, ReTool has shown how "Programming by Demonstration" can be implemented and used to design interactive microtasks with workflows, and that it is possible to design relatively complex crowdsourcing tasks and workflows without programming skills. We hope the work can lead to more intuitive tools for task design so that more people can enjoy the power and convenience of crowdsourcing in the future.

ACKNOWLEDGEMENT

NExT research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

1. Amazon. 2016. Amazon Mechanical Turk Homepage. (2016). https://www.mturk.com.
2. Amazon. 2016. Developer Tools and Software Development Kits (SDKs) of Amazon Mechanical Turk. (2016). https://requester.mturk.com/developer/tools.
3. Paul André, Aniket Kittur, and Steven P. Dow. 2014. Crowd synthesis: extracting categories and clusters from complex data. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing - CSCW '14*. ACM Press, New York, New York, USA, 989–998. DOI: http://dx.doi.org/10.1145/2531602.2531653
4. Michael S Bernstein, Joel Brandt, Robert C Miller, and David R Karger. 2011. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 33. DOI: http://dx.doi.org/10.1145/2047196.2047201
5. M S Bernstein, G Little, R C Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 313–322. DOI: http://dx.doi.org/10.1145/1866029.1866078
6. Jeffrey P Bigham, Samual Samuel White, Tom Yeh, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Robin Miller, Aubrey Tatarowicz, and Brandyn White. 2010. VizWiz: Nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*. 333–342. DOI: http://dx.doi.org/10.1145/1866029.1866080
7. Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. 2010. Visual recognition with humans in the loop. In *Proceedings of the 11th European conference on Computer vision: Part IV - ECCV '10*, Vol. 6314. Springer Berlin Heidelberg, Berlin, Heidelberg, 438–451. DOI: http://dx.doi.org/10.1007/978-3-642-15561-1
8. Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in*

Computing Systems - CHI '13. 1999. DOI: http://dx.doi.org/10.1145/2470654.2466265

9. CrowdFlower. 2016. CrowdFlower Homepage. (2016). https://www.crowdflower.com.
10. Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S. Bernstein, Alex Berg, and Li Fei-Fei. 2014. Scalable multi-label annotation. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 3099–3102. DOI: http://dx.doi.org/10.1145/2556288.2557011
11. Daniel Conrad Halbert. 1984. *Programming by example*. Ph.D. Dissertation. University of California, Berkeley.
12. Kotaro Hara, Vicki Le, and Jon Froehlich. 2013. Combining Crowdsourcing and Google Street View to Identify Street-level Accessibility Problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 631–640. DOI: http://dx.doi.org/10.1145/2470654.2470744
13. Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. 4017–4026. DOI: http://dx.doi.org/10.1145/2556288.2556986
14. Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. 2012. CrowdWeaver. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*. ACM, 1033. DOI: http://dx.doi.org/10.1145/2145204.2145357
15. Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM, ACM Press, New York, New York, USA, 43. DOI: http://dx.doi.org/10.1145/2047196.2047202
16. Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*. ACM, 1003. DOI: http://dx.doi.org/10.1145/2145204.2145354
17. Walter S. Lasecki, Mitchell Gordon, Steven P. Dow, and Jeffrey P. Bigham. 2014. Glance: enabling rapid interactions with data using the crowd. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*. ACM Press, New York, New York, USA, 511–514. DOI: http://dx.doi.org/10.1145/2559206.2574817
18. James R. Lewis. 1995. IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction* 7, 1 (jan 1995), 57–78. DOI: http://dx.doi.org/10.1080/10447319509526110
19. Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. 2010. TurKit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST '10*. ACM, ACM Press, New York, New York, USA, 57. DOI: http://dx.doi.org/10.1145/1866029.1866040
20. Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z Gajos. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 1. DOI: http://dx.doi.org/10.1145/2047196.2047198
21. Hao Su, Jia Deng, and Li Fei-fei. 2012. Crowdsourcing Annotations for Visual Object Detection. In *Proc. AAAI Human Computation' 12*. 40–46.
22. Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *Proceedings of the VLDB Endowment* 5 (aug 2012), 1483–1494. http://arxiv.org/abs/1208.1927