Crowdsourcing processes: a survey of approaches and opportunities

Pavel Kucherbaev[1], Florian Daniel[1,2], Stefano Tranquillini[1], Maurizio Marchese[1]

[1] University of Trento, Italy

[2] Tomsk Polytechnic University, Russia

Abstract. This article makes a case for crowdsourcing approaches that are able to manage crowdsourcing processes, that is, crowdsourcing scenarios that go beyond the mere outsourcing of multiple instances of a micro-task and instead require the coordination of multiple different crowd and machine tasks. It introduces the necessary background and terminology, identifies a set of analysis dimensions, and surveys state-of-the-art tools, highlighting strong and weak aspects and promising future research and development directions.

Keywords: Crowdsourcing, crowdsourcing processes, tools and platforms

Introduction

Crowdsourcing is the outsourcing of a unit of work to a crowd of people via an open call for contributions [1]. Thanks to the availability of online crowdsourcing platforms, such as Amazon Mechanical Turk or CrowdFlower, the practice has experienced a tremendous growth over the last few years [13] and demonstrated its viability in a variety of different fields, such as data collection and analysis or human computation - all practices that leverage on so-called micro-tasks, which ask workers to complete simple assignments (e.g., label an image or translate a sentence) in exchange for an optional reward (e.g., few cents or dollars). The power of crowdsourcing is represented by the crowd, which may be huge and span the World, and its ability to process even thousands of tasks in short time.

The practice is, however, also increasingly struggling with the inherent limitations of crowdsourcing platforms: not all types of work can easily be boiled down to simple micro-tasks, most platforms still require significant amounts of manual work and configuration, and there is only very limited support for structured work, that is, work that requires the integration of different tasks and multiple actors, such as machines, individuals and the crowd. We call these kinds of structured works *crowdsourcing processes*, since they require the coordination of multiple tasks, actors and operations inside an integrated execution logic.

Without proper support for the design and execution of crowdsourcing processes, running them requires a huge amount of manual development, data management and coordination effort as well as specialized expertise. This shortcoming is acknowledged by the recent emergence of advanced crowdsourcing approaches, such as TurKit [8], Jabberwocky [2], CrowdDB [4] and similar, which all aim to ease the development and execution of crowdsourcing processes, typically by building on top of existing crowdsourcing platforms. However, they also all come with a different perspective on the problem and, hence, present different features and capabilities.

The purpose of this article is to introduce the reader to the problem of developing and running crowdsourcing processes and to provide an up-to-date picture of the approaches that have emerged so far. We identify a set of dimensions for the analysis of platforms for crowdsourcing processes and review the state of the art accordingly. The analysis produces a set of considerations that may direct future research and development efforts.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

Crowdsourcing processes

Although not explicitly named as "crowdsourcing processes", literature is rich of examples of scenarios that could benefit from explicit design and runtime support for crowdsourcing processes. For instance:

- Kulkarni et al. [7] crowdsource *article writing* (an article about the attractions of NYC) that involves tasks like structuring an article, writing narrative, splitting content into sections, adding pictures, iterating over content, and coordinating workers that write, correct or structure text.
- Kittur et al. [6] crowdsource a *trip planning* scenario (a road trip from NYC to San Francisco) that requires, for instance, collecting routes, voting for routes, collecting details about hotels, restaurants, attractions, and iterating over the options based on feedback from the crowdsourcer (who crowdsources the micro-tasks; often called "requester").
- Marge et al. [14] study different *audio transcription* experiments (route instructions for robots), which requires, for example, hosting audio records, deploying tasks in different batches, transcribing fragments and gluing them together, iterating over transcriptions until no typos are left, and controlling that workers don't contribute to different batches to avoid learning effects.
- Tranquillini et al. [9] *mine patterns* from models with the help of the crowd, a scenario that requires dedicated task interfaces for the interactive selection of patterns, the coordination of pattern identification and assessment tasks, automatically splitting/aggregating the available dataset, filtering patterns, and similar.
- The Galaxy Zoo project (www.galaxyzoo.org) is a good example of *image classification* process that involves tasks like classifying images into spiral, elliptical, irregular or no galaxy, using a redundant number of workers, describing identified galaxies in function of their galaxy type (e.g., number of arms in a spiral galaxy), and asking experts to resolve possible disagreements.

These examples show that in many practical settings crowdsourcing is not just a matter of deploying a set of simple micro-tasks on a given platform. Instead, it may comprise several different tasks (writing, transcribing, classifying, aggregating, spell checking, voting), actors (crowdsourcers, workers, experts) and automated operations (data splitting, resolving redundancy/multiple delegations, taking decisions about whether to involve an expert or not, synchronizing tasks). Running such processes on top of micro-task crowdsourcing platforms requires significant amounts of manual work, e.g., to split/aggregate datasets or tasks, design task UIs for each task in the process, deploy tasks on the target platform, monitor task executions, collect data, integrate them, split them again, etc. This is highly time consuming and inefficient, and there is huge potential for automation.

Dimensions of analysis

In order to compare the capabilities of existing solutions for the development of crowdsourcing processes, from the above examples we derive the following core dimensions and sub-dimensions:

- ***Definition language***: Developing a crowdsourcing process requires a definition language following some paradigm and notation.
  - *Paradigm*: tells whether the language is imperative, declarative or configuration-based (restricted to predefined templates or patterns).
  - *Notation*: specifies the specific language used, e.g., Scala or BPMN or extensions thereof.
- ***Task support***: Crowd tasks are micro-tasks performed by the workers of the crowd; they leverage on crowd providers and may provide for different crowd management features. Machine tasks are automated operations performed by a machine, e.g., a data transformation.
  - *Crowd provider*: tells which crowd provider (crowdsourcing platform) is supported.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

    - *Crowd management*: tells whether additional crowd management features (e.g., pre-selection or separation of duties) are supported.
    - *Machine task definition:* tells how machine tasks are specified, e.g., via Web services or scripts.
- ***Control flow support***: Automating work means automatically coordinating tasks, that is, controlling the flow of action. The core control flow features crowdsourcing processes may need are:
    - *Task instantiation* (individual and multiple instances);
    - *Sequential execution*;
    - *Parallel execution*;
    - *Decision points* for conditional flows;
    - *Looping/iterating* over similar tasks or data items;
    - *Sub-processes* (or routines/procedures) to support reuse.
- ***Data management support***: Next to progressing the computation from one task to another, it is also mandatory to provide each task with the necessary input data. The basic data management requirements highlighted in our scenario are:
    - *Data hosting*: tells whether the tool hosts data (e.g., audio transcriptions) or references to data (e.g., the URLs to the audio files).
    - *Data passing*: tells whether data are passed via data flows, by value (variables) or by reference (shared memory).
    - *Data splitting/aggregation*: tells how data transformations are specified.
- ***Development support***: Implementing a crowdsourcing process further requires designing suitable crowd tasks and deploying them on the crowdsourcing platform.
    - *Crowd task design*: tells if and how the tool supports the design of crowd tasks.
    - *Task deployment*: tells if and how the tool supports the deployment of tasks.
- ***Quality control support***: Finally, a crucial aspect in crowdsourcing is quality control. This dimension therefore looks at which built-in quality control techniques are supported (e.g., iterating over text until no typos are left).

Approaches and tools

The approaches we review in the following are the result of two years of watching emerging technologies in the context of crowdsourcing. In particular, we consider general-purpose approaches that do not restrict the types of tasks or processes one can crowdsource. Also, at the time of writing, suitable research papers or online resources must have been available to allow us to make an informed assessment of the identified dimensions. These criteria led us to the 11 approaches we describe next.

Selected approaches

Turkit [8] is a JavaScript-inspired scripting language that allows one to programmatically deploy tasks on Mechanical Turk and to pass data among tasks. *AutoMan* [3] is a Scala-based programming language similar to TurKit that automatically manages the scheduling and pricing of task instances and the acceptance and rejection of results, given a target result quality. *Jabberwocky* [2] is a MapReduce-based human computation framework with an parallel programming framework and language. *CrowdComputer* [9] is a BPMN-based design and runtime environment for complex crowdsourcing processes with support for crowd and machine tasks as well as individuals (e.g., experts). *CrowdLang* [11] is a BPMN-inspired modeling language with crowdsourcing-specific constructs. *CrowdWeaver* [5] is a similar model-based tool with a proprietary notation. *CrowdDB* [4] is an SQL-extension that allows one to embed crowd tasks (e.g., inputs and comparisons) into SQL queries. *AskSheet* [12] is a Google Spreadsheet extension with functions that allow the spreadsheet to leverage on crowdsourcing tasks. *Turkomatic* [7] is a

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

crowdsourcing tool for complex tasks that delegates not only work to the crowd but also task management operations (e.g., splitting tasks). *CrowdForge* [6] is a Django-based crowdsourcing framework for crowdsourcing processes similar to Turkomatic that however follows the Partition-Map-Reduce approach. *CrowdSearcher* [10] is a system that allows one to design processes using reusable design patterns and to leverage on machine and crowd tasks as well as on tasks deployed on Facebook.

We are also aware of instruments like CrowdFlow, Quirk, TurkDB, WorkFusion, CrowdFlower Workflows and others, but we were not able to collect enough public information on them. Other approaches, such as CrowdTruth or QualityCrowd2, are tailored to specific domains (collection of gold data for machine learning and video quality assessment, respectively).

Comparison of features

Table 1 describes the selected platforms applying the dimensions and sub-dimensions of analysis introduced earlier. We also add a "public availability" dimension to the analysis, in order to reflect if and how an approach can be tried out and tested. To better highlight commonalities and differences, we group the approaches according to the *paradigm* of their process *definition language* (the sub-order does not follow any temporal or functional order):

- *Imperative, textual*: the crowdsourcer writes code telling *how* the process is executed. Specific *notations* used are Scala, a JavaScript-like language, or a proprietary language (Dog);
- *Imperative, visual*: he models *how* to execute the process visually using graphical abstractions. Concrete notations are BPMN extensions, BPMN-like notations, or custom notations;
- *Declarative*: he defines *what* should be processed or obtained as an output. SQL or spreadsheet formulas are examples of notations used; and
- *Configuration*: he fills configuration properties that set up a *pre-defined* process logic. In this case, the crowdsourcer is typically guided through the configuration by a wizard.

As for the *task support*, the most used *crowd provider* is Mechanical Turk (MTurk), which is however restricted to crowdsourcers from the USA only; CrowdFlower does not have this restriction. Some approaches self-manage an own crowd. CrowdSearcher proposes an alternative interpretation and also supports deploying tasks on Facebook, which adds extra opportunities like access to people that would not use conventional crowdsourcing platforms (e.g., teenagers) and volunteer work by people in the crowdsourcer's own social network. CrowdComputer, given its roots in business process management (BPM) that focuses on the coordination of human work, also supports assigning tasks to individuals (e.g., an expert) via a conventional BPM engine. *Crowd management* features are only scarcely supported and mostly focus on worker pre-selection, bonus payments and approval/rejection of results. *Machine tasks* come in very different flavors: the imperative, textual approaches allow the crowdsourcer to write own scripts, the visual approaches rather support reusable modules like Web services, the declarative approaches are limited to their environment's native capabilities, and the configuration approaches may provide for customizable, built-in machine tasks, e.g., for data management.

From the *control flow* perspective, all the platforms support automated *task instantiation*. Given their imperative nature, both the textual and the visual approaches support most of the control flow features; control flow support by the declarative and configuration approaches is platform specific. *Sequential execution* is supported by all except AskSheet (spreadsheet functions are evaluated in parallel). *Parallel execution* is more platform specific. *Decision points* come either as if-statements in imperative, textual approaches and AskSheet, graphical gateways in the imperative, visual approaches, or adaptation rules in CrowdSearcher. There are no explicit decision points in Turkomatic, where the workers decide at runtime whether to split a task or to execute it. *Iterative execution* is not supported in platforms without decision points. *Sub-processes* are only weakly supported; if supported, they are either reusable functions (imperative, textual approaches), BPMN processes (CrowdComputer), or Python scripts (CrowdForge).

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

Table 1. *Analysis of crowdsourcing platforms for crowdsourcing processes*

		TurKit	AutoMan	Jabberwocky	CrowdComputer	CrowdLang	CrowdWeaver	CrowdDB	AskSheet	Turkomatic	CrowdForge	CrowdSearcher
Definition language	paradigm	imperative, textual	imperative, textual	imperative, textual	imperative, visual	imperative, visual	imperative, visual	declarative	declarative	declarative	configuration	configuration
	notation	JavaScript-like	Scala	Dog	BPMN extension	BPMN-like	custom modeling language	extended SQL	Google Spreadsheet formula	–	wizard for config, Python for custom processes	wizard plus adaptation rules
Task support	crowd provider	MTurk	MTurk	self	self, BPM engine	MTurk	CrowdFlower	MTurk	MTurk	MTurk	MTurk	MTurk, Facebook
	crowd management	–	approvement and rejection	profile-based (expertise, demographics, groups) preselection	profile-based preselection	–	–	approvement, rejection, bonus payment	–	–	–	–
	machine tasks definition	script	script	script	generic Web services	generic machine tasks	generic machine tasks	SQL operations	spreadsheet functions	–	–	data management operations
Control flow support	task instantiation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	sequential execution	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓
	parallel execution	-	-	✓	✓	✓	✓	–	✓	✓	✓	✓
	decison points	✓	✓	✓	✓	✓	–	–	✓	–	–	✓
	looping / iterative execution	✓	✓	✓	✓	✓	–	–	–	–	–	✓
	sub-process	✓	✓	✓	✓	–	–	–	–	–	✓	–
Data management support	data hosting	data	data	data	references	data	data	data	data	data	data	data
	data passing among tasks	by value	by value	by value	by reference	by value	data flow	data flow	by reference	self-managed data flow	by value	data flow
	data splitting, aggregating	script	script	script	built-in	built-in	built-in	SQL operations	spreadsheet functions	by crowd	by crowd	built-in
Development support	task design	manual	pre-defined	manual	manual	automatic	wizard	automatic	wizard	pre-defined	manual	wizard
	task deployment	automatic	automatic	automatic	automatic and manual	automatic	automatic	automatic	automatic	automatic	automatic	automatic
Quality control support		voting	confidence levels under given budget	–	custom logics	–	control questions, consensus	consensus	rating, consensus	voting	voting	consensus
Public availability		open-source	open-source	–	open-source, deployed online	–	–	–	–	–	open-source	deployed online

Regarding *data management*, all platforms support the *hosting* of data, except CrowdComputer that only manages data references. While this requires the crowdsourcer to manage the actual data himself, it reduces data transfer and allows the crowdsourcer to protect data that is sensitive (e.g., images with nudity) or subject to local regulations (e.g., healthcare data). CrowdLang and CrowdForge pass data by value; CrowdComputer and AskSheet pass data by reference; the other approaches use direct data flows. Data *splitting* and *aggregating* logics are either built-in operators, custom crowd tasks, or coded in the underlying process definition language.

Development support for *task design* comes in different flavors: manual, automatic, wizard-based, or pre-defined tasks. Manual design asks the crowdsourcer, for instance, to develop HTML-based Web forms (CrowdComputer) or XML task definitions (CrowdForge). AutoMan is instead able to automatically generate task user interfaces out of an SQL query and the affected table schemas. A wizard-based design is proposed by CrowdSearcher, while Turkomatic and AutoMan are examples of platforms that support only pre-defined text editing and voting tasks. *Task deployment* is generally automatic; CrowdComputer asks the crowdsourcer to host task implementations, which may require also a manual intervention.

Also for *quality control*, four main approaches can be identified: *rating* (a crowd task is used to rate work of another task); *voting* (a crowd task is used to collect preferences for results of another task); *consensus* (new results are accepted until at least two or more results match); and *control questions* (extra questions, for which the correct answer is known, are injected into a crowd task to evaluate a worker). Automan stands out in this context: it allows the crowdsourcer to define an overall budget and a target confidence level for the results and automatically manages the necessary pricing, approval and rejection of tasks.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

As for the *availability* of the approaches, 4 out of 11 platforms are open-source projects, but only 2 are actually deployed online and ready for use; 6 platforms are not available at all. In this respect, it interesting to note that all the approaches are research prototypes. We are aware that companies like CrowdFlower and Workfusion deploy and run crowdsourcing processes on behalf of their enterprise customers at a daily basis. CrowdFlower Workflows, CrowdFlower's internal platform for crowdsourcing processes, is also available for enterprise customers; however, the commercial offering is still very limited, if not inexistent. This may be an indication that the goals and effectiveness of platforms for crowdsourcing processes are not yet clear and crisp enough for the market.

Discussion and outlook

The selection of crowdsourcing approaches discussed in this article shows that there already exists a diverse and growing ecosystem of even sophisticated solutions. As usual with automation instruments, their usefulness in practice is a trade-off between how often a process is repeated (e.g., to test different crowdsourcing settings) and how easy it is to use the instrument (e.g., compared to manual crowdsourcing). If not in their current form (stand-alone platforms), we however expect that – after the initial prototypes introduced in this article – support for crowdsourcing processes will percolate into and enhance existing crowdsourcing platforms, e.g., as is already happening with CrowdFlower Workflows.

We further discussed our analysis with Lukas Biewald, CEO of CrowdFlower (the company operates as both crowd provider and crowdsourcer on behalf of its key customers), so as to jointly identify some of the challenges the crowdsourcing community will have to approach next in order to foster tools for crowdsourcing processes:

- *Integration*: The prevalence of proprietary notations for process definition risks to make integration with other computing environments cumbersome. Especially the textual approaches (except AutoMan: Scala) are hard to integrate into other programming environments; the same holds for the visual approaches (except for CrowdComputer: BPMN) and the configuration-based approaches. Only the declarative approaches seem well integrated into their host environments (databases and spreadsheets). However, many of the surveyed approaches are equipped with APIs that can be programmed and leveraged on for integration from the outside.
- *Quality control*: The supported techniques to control the quality of the results produced by the crowd are still rather limited, and quality is controlled at the granularity of individual crowd tasks only. More complex quality control logics (e.g., providing quality guarantees able to raise exceptions and to dynamically compensate for low quality) or logics that control quality at the granularity of entire crowdsourcing processes (e.g., able to maximize the quality of outputs while, at the same time, keeping a given budget and time restrictions) still require more research.
- *Adaptive process execution*: Crowdsourcing usually requires a significant testing and fine-tuning effort for both individual tasks and entire processes. Many times, processes are constructed by running a task, analyzing its outputs, deciding whether a post-processing of the data is needed or whether the next crowd task can be executed, etc. This, on the one hand, asks for novel testing techniques for crowdsourcing processes and, on the other hand, for crowdsourcing processes that can be started even if not yet completely defined and that can be refined at runtime, e.g., by adding ad-hoc tasks or operations.
- *Worker selection and training*: The success of crowdsourcing depends first and foremost on the quality of the work produced, and this, in turn, depends on the skills and abilities of the workers. Not always is the solution selecting workers with the necessary skills, e.g., if a given skill or domain knowledge is not present at all. A challenge for future crowdsourcing practice is therefore understanding how to train workers for specific skills, how to motivate them to participate in training, how to reward and certify training, and how to properly value training in the selection of workers – all advanced crowd management aspects that will ask for effective answers.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

This survey is based on the analysis of research papers and hands-on tests of the available prototypes. Acknowledging the limitations of this approach (the level of detail of papers, impossibility to access prototypes, the pace of evolution), we intend to add a new section on crowdsourcing processes to the "Crowdsourcing" entry in Wikipedia (https://en.wikipedia.org/wiki/Crowdsourcing), enabling everybody to integrate and extend this analysis as a community effort.

Acknowledgements. We thank Lukas Biewald (CrowdFlower), Nicola Sambin (SpazioDati), Patrick Minder and Abraham Bernstein (CrowdLang), Alexander J. Quinn (AskSheet), and Marco Brambilla (CrowdSearcher) for their help.

References

1. J. Howe. "Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business," Crown Publishing Group, 2008.
2. S. Ahmad, A. Battle, Z. Malkani, and S. Kamvar. "The Jabberwocky programming environment for structured social computing," *UIST 2011*, 53–64.
3. D. W. Barowy, C. Curtsinger, E. D. Berger and A. McGregor. "AutoMan: A Platform for Integrating Human-based and Digital Computation," *OOPSLA 2012*, 639–654.
4. M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh and R. Xin. "CrowdDB: Answering Queries with Crowdsourcing," *SIGMOD 2011*, 61–72.
5. A. Kittur, S. Khamkar, P. Andre and R. Kraut. "Crowdweaver: visually managing complex crowd work," *CSCW 2012*, 1033–1036.
6. A. Kittur, B. Smus, S. Khamkar and R. E. Kraut. "Crowdforge: crowdsourcing complex work," *UIST 2011*, 43–52.
7. A. Kulkarni, M. Can, and B. Hartmann. "Collaboratively crowdsourcing workflows with turkomatic," *CSCW 2012*, 1003–1012.
8. G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. "Turkit: human computation algorithms on Mechanical Turk," *UIST 2010*, 57–66.
9. S. Tranquillini, F. Daniel, P. Kucherbaev and F. Casati. "Modeling, Enacting and Integrating Custom Crowdsourcing Processes," *ACM Transactions on the WEB*, 2015, conditionally accepted.
10. A. Bozzon, M. Brambilla, S. Ceri, A. Mauri and R. Volonterio. "Pattern-based specification of crowdsourcing applications," *ICWE 2014*, 218–235.
11. P. Minder, A. Bernstein. "CrowdLang - Programming Human Computation Systems," *WebSci 2012*.
12. A. J. Quinn and B. B. Bederson. "AskSheet: efficient human computation for decision making with spreadsheets," *CSCW 2014*, 1456-1466.
13. Crowdsourcing Week. "2014 Global Crowdsourcing Pulsecheck: 1st Annual Survey Topline Results," http://www.slideshare.net/crowdsourcingweek/2014-global-crowdsourcing-pulsecheck-1st-annual-survey-topline-results" (April 2015).
14. M. Marge, S. Banerjee, A. I. Rudnicky. "Using the Amazon Mechanical Turk for transcription of spoken language," *ICASSP* 2010, 5270-5273.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE

Biographies

Pavel Kucherbaev is a PhD student in the Department of Information Engineering and Computer Science at the University of Trento, Italy, and the European Institute of Technology Digital program. His research focuses on quality control and timeliness of crowdsourcing micro-tasks. Pavel is also interested in space exploration and entrepreneurship. Contact him at pavel.kucherbaev@unitn.it.

Florian Daniel is a senior research fellow at the University of Trento, Italy, and professor at the Tomsk Polytechnic University, Russia. He received his PhD from Politecnico di Milano. His research focuses on crowdsourcing, web engineering, mashups, service-oriented computing and business process management. Florian is also passionate about running, mountains and beer brewing. Contact him at florian.daniel@unitn.it.

Stefano Tranquillini is a post-doctoral researcher in the Department of Information Engineering and Computer Science at the University of Trento, Italy, where he received his PhD. His main research interests are in the area of business process management, crowdsourcing and the integration of the two. He also plays futsal and is fond of photography. Contact him at stefano.tranquillini@unitn.it.

Maurizio Marchese is an associate professor in the Department of Information Engineering and Computer Science at the University of Trento, Italy, and a Director of Education in the European Institute of Technology ICT Labs initiative for the Trento node. His main research interest are in social informatics where he studies how information systems can realize social goals, apply social concepts, and become sources of information relevant for social sciences and for analysis of social phenomena. Contact him at maurizio.marchese@unitn.it.

Digital Object Indentifier 10.1109/MIC.2015.96 1089-7801/$26.00 2015 IEEE