# A Survey of Task-Oriented Crowdsourcing

Nuno Luz[1]

*GECAD (Knowledge Engineering and Decision Support Group), Polytechnic of Porto – School of Engineering*

nmalu@isep.ipp.pt

GECAD, R. Dr. António Bernardino de Almeida 431, 4200-072, Porto, Portugal

Nuno Silva

*GECAD (Knowledge Engineering and Decision Support Group), Polytechnic of Porto – School of Engineering*

nps@isep.ipp.pt

GECAD, R. Dr. António Bernardino de Almeida 431, 4200-072, Porto, Portugal

Paulo Novais

*CCTC (Computer Science and Technology Center), University of Minho*

pjon@di.uminho.pt

University of Minho, CCTC/DI, Campus of Gualtar, 4710 -057, Braga, Portugal

**Abstract:** Since the advent of artificial intelligence, researchers have been trying to create machines that emulate human behaviour. Back in the 1960s however, Licklider (1960) believed that machines and computers were just part of a scale in which computers were on one side and humans on the other (human computation). After almost a decade of active research into human computation and crowdsourcing, this paper presents a survey of crowdsourcing human computation systems, with the focus being on solving micro-tasks and complex tasks. An analysis of the current state of the art is performed from a technical standpoint, which includes a systematized description of the terminologies used by crowdsourcing platforms and the relationships between each term. Furthermore, the similarities between task-oriented crowdsourcing platforms are described and presented in a process diagram according to a proposed classification. Using this analysis as a stepping stone, this paper concludes with a discussion of challenges and possible future research directions.

*Crowdsourcing, Human Computation, Survey, Complex Tasks, Micro-task*

[1] Corresponding author

# Introduction

Since the advent of artificial intelligence, researchers have been trying to create machines that emulate human behaviour. This has led to multiple branches of artificial intelligence such as multi-agent systems, reasoning and negotiation. Back in the 1960s however, Licklider (1960) believed that machines and computers were just part of a scale which weights humans on one side, and computers on the other. His vision was that computers and humans should work together performing complementary roles (Licklider 1960; Quinn and Bederson 2011).

It was only recently that relevant research emerged and brought humans into computer affairs. The early contributors to this retake on Lickliders' vision might as well be the social Web and the harnessing of collective intelligence (Gruber 2008). These proved that humans have great complementary abilities that are relative to computers, and that they can act as guided computational units. As a part of collective intelligence (Quinn and Bederson 2011), human computation (Von Ahn 2009) re-emerged as a relevant research field. Shortly after, the term crowdsourcing was coined (Howe 2006), leading to yet another field of research that is highly connected to human computation.

After almost one decade of active research into human computation and crowdsourcing, several approaches and business models based on crowdsourcing have emerged, managing and distributing work to the crowd (Doan et al. 2011; Quinn and Bederson 2011; Yuen et al. 2011a). In this sense, crowdsourcing (currently the most popularized term) can be seen from two different perspectives: a business domain-specific perspective, and a technical domain-independent perspective.

From a business perspective, there is great interest in accomplishing specific business tasks efficiently and effectively in terms of time and monetary costs. Studies in this context tend to focus on the domain-specific details of the task, giving special concern to user motivation and quality-control aspects.

A technical and domain-independent perspective, on the other hand, puts the emphasis on methods, techniques and frameworks for solving problems (in general) that machines alone aren't yet able to solve (Von Ahn 2009; Quinn and Bederson 2011). In some cases, problem-solving can be achieved by replacing machines with humans in certain computation steps where humans usually per-

form better (human computation). In this context, the focus is on the creation and deployment of mechanisms that efficiently and effectively facilitate the crowdsourcing and human computation process.

So far, most work on providing a survey and classification of crowdsourcing systems (Doan et al. 2011; Quinn and Bederson 2011; Yuen et al. 2011a) adopted a business and user perspective.

This paper presents a survey of crowdsourcing human computation systems from a technical domain-independent perspective, with a focus on solving micro-tasks and complex tasks. First, a discussion of the concepts collective intelligence, human computation and crowdsourcing is presented, followed by their application and role in several different domains. Afterwards, a detailed analysis and systematization of several micro-task and complex task crowdsourcing systems is presented, along with their comparison, terminologies and process. This analysis is given according to scientific publications and the empirical analysis of the online system, if available. An ontology that allows the classification of such systems according to different technical aspects is proposed. Finally, current challenges and possible future approaches are discussed.

## Human Computation and the Wisdom of Crowds

Humans are innately social and the intrinsic aspects of human cooperation have been the subject of great research efforts (Porter 2008). We, humans, not only tend to form a clustered structure of relationships (social circle), but also extract individual benefits from them (Levine and Kurzban 2006). Social circles have a great impact on our lives, influencing our ideas and behaviour (Konstas et al. 2009). Not so long ago, the information that a person had access to was mostly the information flowing inside his social circle. Nowadays, our social circle also acts as a filter for all the vast amounts of information and choices delivered to us every day (Ma et al. 2011) by other means (e.g. social media, internet).

The emergence of the social web has brought new powerful web applications that connect people on a global scale, and allow them to reap the benefits of social life from online virtual environments in a global scale. Along with their huge popularity, online social networks allow the retrieval of significant amounts of important social data, which can be used to promote social benefits (Levine and Kurzban 2006).

One of the most straightforward benefits we extract from society comes from asking our friends for an opinion or advice (Ma et al. 2011). It is possible to apply a similar mechanism to online social networks by automatically filtering data, and providing the user with relevant and personalized results according to the opinions coming from his online social circle. The difference is that, unlike humans alone, the introduction of machines allows that procedure to be performed for millions of items, covering a wide social circle.

Online social networks also destroy geographical barriers, thus promoting the combination of behaviours and ideas on a global scale (Porter 2008). This combination is often referred to as collective intelligence (Luo et al. 2009).

An interesting example of the importance of collective intelligence is what Porter (2008) regards as the Amazon Effect. To explain the Amazon Effect, he describes a usability study where people were asked to buy a product at a certain online store. A lot of people wanted to go to Amazon first, and when they were asked why, they just answered that they would like to do some research on the product, even if they were not buying it on Amazon.

Brabham (2008a) follows Surowieckis' (2004) view on the wisdom of crowds (often referred to in the context of the Web as collective intelligence), which states that it emerges from aggregating individual solutions, instead of averaging them (as in the case of Amazon's review system). This view is particularly relevant in the context of problem solving, where aggregating individual solutions often leads to a better solution than any of the best originally proposed individual solutions. Following this view, Brabham argues that crowdsourcing is a model achieved through the Web that is "capable of aggregating talent, leveraging ingenuity while reducing costs and time formerly needed to solve problems". Crowdsourcing is a term popularized by Howe (2008; 2006) that emerged in the context of a paradigm shift in business models. This shift originated from companies that started to provide outsourcing services relying on anonymous communities or crowds throughout the Web (e.g. iStockPhoto[2], InnoCentive[3] and Amazon's Mechanical Turk[4]). By 2006, these communities were growing into incredi-

[2] http://www.istockphoto.com

[3] http://www.innocentive.com

[4] https://www.mturk.com

ble valuable work forces capable of performing several specific tasks in exchange for small monetary rewards.

Since then, several similar definitions for the term crowdsourcing have been given. According to Howe (2008), "crowdsourcing is the act of taking a task traditionally performed by a designated agent (such as an employee or a contractor) and outsourcing it by making an open call to an undefined but large group of people". Doan et al. (2011) define crowdsourcing as a system that "enlists a crowd of humans to help solve a problem defined by the system owners, and if in doing so, it addresses" the challenges of recruiting and retaining users, defining which contributions can be made by users, combining these contributions and evaluating user performance.

Quinn and Bederson (2011) not only provide a definition for crowdsourcing, but also compare it to terms such as human computation, social computing and collective intelligence. They aggregate several definitions found in literature and state that crowdsourcing is a form of collective intelligence that overlaps human computation.

The term human computation dates back to the early years of artificial intelligence, in the 1960s, where it was envisioned that computers and humans should work together performing complementary roles (Licklider 1960; Quinn and Bederson 2011). Still, the vision of human-computer collaboration only started to be properly explored after 2005, the year von Ahn published his doctoral thesis entitled Human Computation (Von Ahn 2009). Von Ahn proposes the use of human algorithm games to harness the distributed processing power of humans to perform specific tasks. In accordance, human computation can be defined as a computational process that involves humans and their cooperation in order to solve problems that computers cannot yet solve (Quinn and Bederson 2011). This definition is complemented by stating that "human computation does not encompass online discussions or creative projects where the initiative and flow of activity are directed primarily by the participants' inspiration, as opposed to a predetermined plan designed to solve a computational problem".

Quinn and Bederson (2011) argue that while human computation requires humans to act as managed units that merely perform a computation, crowdsourcing requires several humans to cooperate in a process by performing a computation or a creative task that is not always managed by computers (e.g. Wikipedia).

Crowds have an important role in human computation. Besides providing good amounts of computational power, applicable in tasks that machines can barely solve with efficiency and efficacy, they can also be used for redundancy.

With the evolution and absorption of crowdsourcing and human computation into market-places and businesses, it can be observed that while human computation (HC) is a term that is mostly used by the scientific community, crowdsourcing (CS) is a term highly employed in the business world.

Throughout this paper, the terms micro-task, task and complex task will be heavily employed. In order to reduce ambiguity fig. 1 defines the relationships between each concept, considered in the context of this paper.

Two types of tasks are considered: micro-tasks and complex tasks. While micro-tasks are atomic computation operations, complex tasks are (possibly ordered) sets of micro-tasks (e.g. workflows of micro-tasks) with a specific purpose.

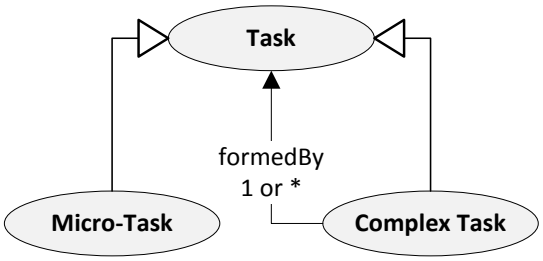


**Fig. 1 - Relationships between the concepts Micro-Task, Task and Complex Task. Both Micro-Task and Complex Task are sub-concepts of Task.**

## Role and Application Domains

Several experiments in different domains have shown that CS and HC have great potential for solving large scale problems that are often difficult for computers to solve automatically, on their own (Von Ahn 2009). These problems usually require a degree of creativity or just common sense plus some background knowledge (Chklovski 2003; Singh et al. 2002). The interpretation and recognition of images and natural language are two examples of these kinds of problems. One of the applications of CS lies in harnessing geographical information. Recently, the production of geo-referenced data, maps, and atlases has moved from mapping agencies and corporations to non-expert users (Goodchild and Glennon 2010). Some of the services that allow this include Flickr, Google's MyMaps, OpenStreetMap, and Wikimapia. Following this trend, Goodchild and Glennon (2010) discuss the applications of CS to the harnessing of geographical infor-

mation for disaster response. They argue about the importance of quality in harnessing geographical information and present an analysis of non-expert user generated geographical information from occurrences of wildfires in Santa Barbara, California. Although further research is needed, there is great potential for quickly generating and spreading disaster-related information through a CS system.

The potential and importance of CS in harnessing geographical information has also been successfully noted and put into practice by Safecast[5], a project that emerged one week after the earthquake that led to the Fukushima Japanese nuclear accident. Safecast is a "global sensor network for collecting and sharing radiation measurements to empower people with data about their environments". In order to collect data, different types of radiation sensors are distributed through volunteers that later use them to collect geo-referenced radiation measurements during their travels. The results are collected and published by Safecast, which provides free access to the data.

Several CS-based businesses have emerged since the advent of CS platforms. While some maintain their own community of workers (e.g. MicroWorkers[6], ShortTask[7]), others interact with one or more CS platforms (e.g. CrowdFlower[8]) offering their services in designing and managing projects and tasks, and obtaining reliable results. Brabham (2008a) discusses several successful applications of CS as business models, which include Threadless, InnoCentive, and iStockPhoto. Threadless crowdsources the design process of t-shirts by promoting online competition. InnoCentive has a different focus, as it crowdsources the research and development of scientific problems as challenges. The last, iStockPhoto, sells photographs, animation, and video clips produced by its crowd of artists. Interestingly, surveys of the iStockPhoto crowd show that the main motivations behind their time and effort are not only monetary but also enjoyment and the development of individual skills (Brabham 2008b).

In 2010, Dawson and Alexandrov published a diagram of the landscape of CS[9]. They distinguish CS systems through thirteen categories, enumerating sever-

[5] http://blog.safecast.org/

[6] https://microworkers.com

[7] http://www.shorttask.com

[8] http://crowdflower.com

[9] http://crowdsourcingresults.com/competition-platforms/crowdsourcing-landscape-discussion

al domains where CS has been applied. The presented categories are: crowdsourcing aggregators (e.g. CrowdFlower), content markets (e.g. iStockPhoto), prediction markets (e.g. Crowdcast), question answering (e.g. Yahoo! Answers), innovation prizes (e.g. XPrize), service marketplaces (e.g. Freelancer), distributed innovation (e.g. InnoCentive), crowdfunding (e.g. KickStarter), competition platforms (e.g. 99 Designs), content-rating (e.g. Delicious), idea platforms (e.g. IdeaScale), data sharing (e.g. Dead Cellzones), reference content (e.g. Wikipedia), cycle sharing (e.g. SETI@Home) and micro-tasks (e.g. Mechanical Turk and ShortTask). Among the application domains featured by these CS systems are business ideas, 3D and graphic design, data analysis, research, tagging, translation, writing and editing, reviewing and software development.

From all the CS systems and common types of CS tasks enumerated by Dawson and Alexandrov, only some qualify as HC systems. This is the case for CS systems under the micro-tasks category. However, although CS systems like Mechanical Turk and ShortTask provide a platform for building any type of tasks, some specific types of tasks have become widely popular for being particularly adequate for micro-task representation, and for being easily accepted by workers. These specific types of tasks are often (as presented in CloudCrowd[10]) writing, editing, categorization, searching, data entry and translation tasks.

In this sense, most CS micro-task systems feature the creation of task templates that can be used to request multiple similar tasks. These systems often provide a predefined set of templates for commonly requested types of tasks. Some of the predefined templates provided by Mechanical Turk and ShortTask include:

- Categorization, classification;
- Data verification (e.g. provide correct spelling)
- Data extraction (e.g. finding a website address);
- Moderation and tagging of multimedia content (e.g. tagging images or videos with adult content);
- Transcription from multimedia content (e.g. audio, video and images);
- Sentiment analysis and surveys;
- Search relevance (e.g. evaluate relevance of search results).

[10] http://www.cloudcrowd.com

Around these CS systems that manage their own community of workers, CS-oriented businesses have started to emerge. Although these businesses tend to provide services with a tendency towards solving complex tasks, they still share many similarities with common CS micro-task system templates. For instance, MobileWorks[11], a CS-oriented company, groups its services into categories such as digitalization of documents, categorization and classification, researching, and harnessing feedback (e.g. through surveys).

Some of these application domains can be easily modelled and managed with single and independent CS micro-tasks. Recently, however, a special interest in employing CS towards solving more complex tasks has emerged (Ahmad et al. 2011; Kittur et al. 2011; Kulkarni et al. 2011; Little et al. 2010; Luz et al. 2012; Sarasua et al. 2012). This interest has led to several approaches being built upon workflows of micro-tasks. The modelling of such workflows allows the CS of a new kind of more complex tasks (e.g. selecting and buying a video camera).

## Quality Control and Evaluation

Over the years, several researchers have dedicated their efforts to evaluating the quality of the results obtained through crowdsourcing tasks (Brabham 2008b; Goodchild and Glennon 2010; Kittur et al. 2008; Paolacci et al. 2010; Willett et al. 2012).

Kittur et al. (2008) present an analysis of crowdsourcing user evaluations using Mechanical Turk. These user evaluations include "surveys, usability tests, rapid prototyping, cognitive walkthroughs, quantitative ratings, and performance measures". By performing two different experiments they concluded that although promising, special care is required when formulating user evaluation micro-tasks. In the first experiment, workers were asked to rate the quality of a Wikipedia article according to questions formulated from the Wikipedia article guidelines. Results showed that nearly 48.6% of the answers were invalid and 30.5% were given in less than one minute. In the second experiment, questions that guided the worker through the article evaluation process (e.g. how many references and images does the article have?) were added to the micro-task. This resulted in a significant

[11] https://www.mobileworks.com

reduction of invalid responses (around 2.5%) and in an increase of the micro-task execution time.

Willett et al. (2012) discuss the quality of results in social data analysis tasks. This kind of task focuses on harnessing explanations and interpretations of data, thus requiring diverse justified analytical answers. The CS process (see fig. 2) starts after the selection of a set of charts by an analyst. An analysis micro-task is then created, in which CS workers provide explanations for each chart. The analysis is followed by a rating micro-task, where workers assess the quality of the previously submitted explanations according to relevance, clarity and plausibility. The results of this task aid the analyst in filtering low quality explanations.

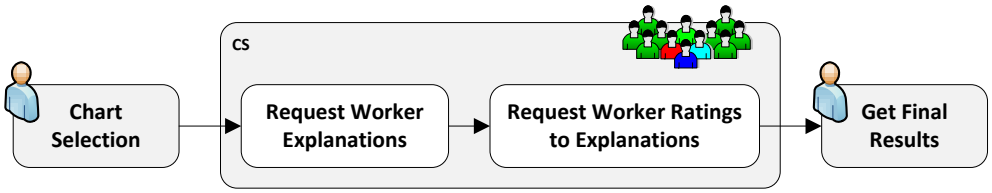


**Fig. 2 - The CS process of data analysis proposed by Willet et al. (2012).**

Implementing this workflow of micro-tasks, Willett et al. (2012) describe seven strategies to improve the quality of explanations, tackling issues like irrelevant, unclear and speculative explanations, inattention to chart detail, and lack of diversity. These strategies are to i) use feature-oriented prompts, ii) provide good examples, iii) include reference gathering subtasks, iv) include chart reading subtasks, v) include annotation subtasks, vi) use pre-annotated charts, and vii) elicit explanations iteratively.

Experiments by Willett et al. show that around 63% of the given explanations are good, and that the described strategies significantly improve the quality of explanations.

Although quality evaluation and control is often a highly domain-specific process, Ipeirotis et al. propose an algorithm that evaluates the quality of workers for general tasks in Mechanical Turk and attributes a score to each worker (Ipeirotis et al. 2010). Following the assumption that biased workers can still provide relevant (although often considered wrong) answers, the algorithm tries to distinguish between error and bias in worker answers.

# Systematization of Task-Oriented Systems

This section presents an analysis and systematization of task-oriented CS and HC systems. A brief description of all the entities present in these systems is given, along with the relationships between them. Afterwards, an abstraction of task-oriented CS and HC processes is presented. This abstraction encompasses most systems and includes complex task-oriented systems based on task workflows. Finally, a description of each of the analysed CS task-oriented systems is provided.

## Entities and Relationships

Throughout the following analysis, several terms will be employed according to different terminologies. These terminologies refer to the entities that are present in the CS and HC process. For this description and systematization the following terms and entities are considered:

- Worker – a person that solves tasks;
- Community – a set of workers;
- Requester – an entity that submits jobs;
- Job – a complex task or workflow of tasks;
- Task – the specification of a task or micro-task, which may be instantiated a multiple number of times;
- Unit – an instance of a task;
- Reference Unit – an instance of a task with a known answer;
- Assignment – an assignment of a unit to a single worker;
- Answer – the given solution of a worker to a specific assignment;
- Qualification – a validated worker skill or expertise in a specific domain;
- Credibility – a measurement of worker performance in completed assignments;
- Workflow – the continuity of work by passing the output of one task as the input of another.

Regardless of their application domain, several subtypes of tasks were identified:

- Partition Task – a task that consists in the partitioning of a complex task into a workflow of simpler tasks;

- Aggregation Task – a task that consists in the aggregation of multiple answers given to another task or job;
- Qualification Task – a task that must be successfully completed in order to obtain a qualification;
- Grading Task – a task that consists of assessing the results of qualification tasks and usually given to highly credible and qualified workers.

The relationships between these entities are represented in the relational diagram in fig. 3. Notice that this diagram represents a generalization and a conceptual model of the analysed systems. It does not take into account implementation details.

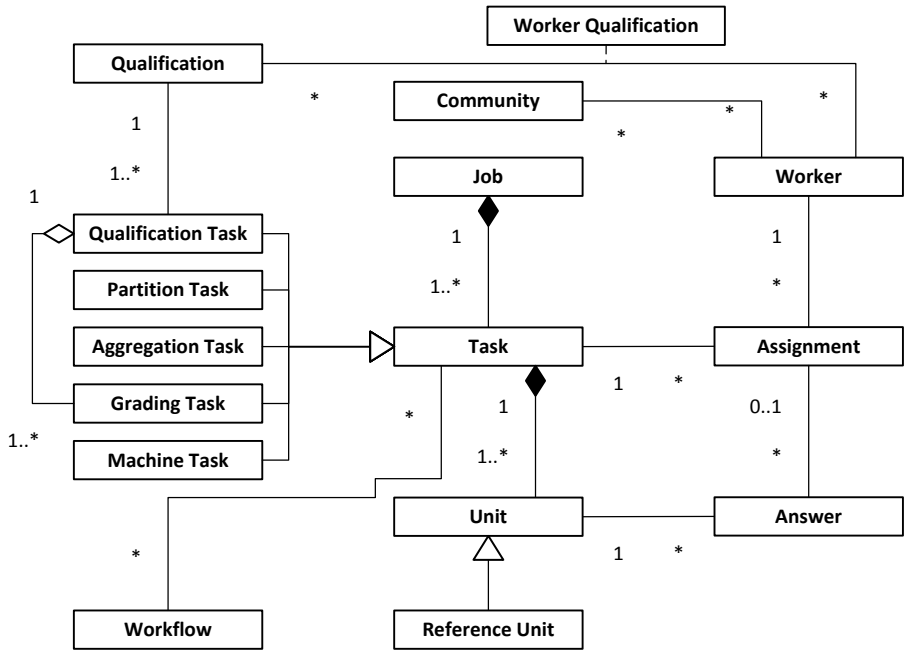


**Fig. 3 - Entity relationship diagram of a common (complex) task CS and HC system.**

## The Process

There are three active entities in a task-oriented CS and HC process: the requester, the worker, and the system itself. Each of these entities has a different role in the process. For instance, the worker only has to select a task, solve it, and later receive (or not) a reward according to his performance. The requester and system flow of actions, however, is more complex. In fig. 4, the whole process is depicted for each of the three active entities. Notice that when solving complex human computation tasks, current systems focus on managing workflows of simple tasks.

The overall process has three phases: the design phase, the online phase, and the conclusion phase. For each of these phases, the job can be found in different states. In the case of the design phase, the job is always in the "not ordered" state. During the online phase, the job may be either "running" or "paused". Finally, during the conclusion phase, the job reaches its' final state, which can be either "finished" or "cancelled".

The *design phase* is an exclusive interaction between the requester and the system, where the requester must configure the job, design each task, and build the task workflow.

In the *online phase* the job is set to run. It is during this phase that the worker must act and solve the task. The requester, on the other hand, can pause the job to modify its configuration and change parts of the workflow (as in the crash and rerun strategy), resuming its execution afterwards.
Just like the job, each worker task in the workflow has an associated state. Before it is reached, the worker task is set as "not ordered". When its execution starts, the worker task is set to "running" with the possibility of being "paused". Finally, it can be either "cancelled" or "finished".

Five main steps are performed during the execution of a worker task: (i) task distribution and worker selection, (ii) assignment to workers, (iii) assignment assessment, and optionally, (iv) result aggregation and (v) worker rewarding. The first three steps are executed in a loop, where the task may be re-assigned to a specific group of workers in the same community. Also, for each task, multiple assignments can be given to several workers, each requiring an assessment. When all the required assignments are solved, workers can be optionally rewarded automatically according to the results in the assignment assessment. All assignments of the same task may also be aggregated in order to provide a final answer.

For each task in the workflow, the previously described process is followed. More specifically, these tasks can run either concurrently, for parallel tasks in the workflow, or sequentially. If the output of two parallel tasks is required as the input of the following task, the system must synchronize and wait for both parallel tasks to be completed before advancing to the next task.

After the workflow is finished, the *conclusion phase* starts, in which the requester can review and reward workers according to their performance. The results of the execution of the workflow are also made available to the requester.

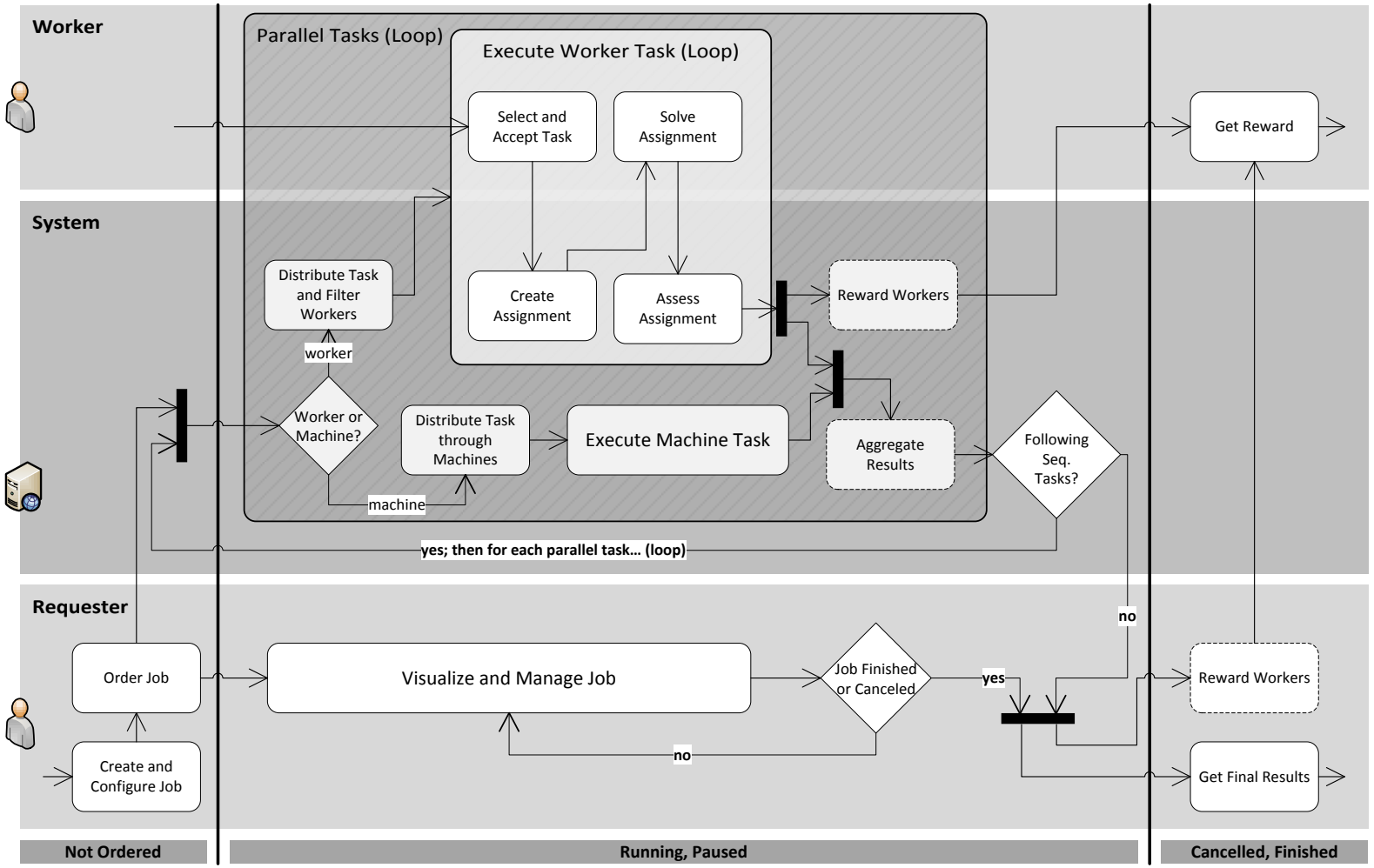


**Fig. 4 - Process diagram of a common (complex) task CS and HC system. Dashed steps may, or may not, exist in different systems.**

## Systems and Platforms

The following analysis contains CS systems that overlap with HC. A special focus is given to the CS of complex tasks. For some of the systems, an empirical study was performed since there are no available publications describing them. In other cases, the system implementation was not available, leading to a study based only on related publications.

### *Mechanical Turk*

Mechanical Turk is an online labour market and pioneer of the CS platforms that specialize in micro-tasks. Each requester (employer) can create HITs (Human Intelligence Tasks) or units of work, which represent multiple micro-tasks that will be executed by workers (employees) in exchange for a small monetary reward (Paolacci et al. 2010).

Requesters can specify several qualification requirements for workers. These requirements often contain test forms that workers must complete in order to assess their qualifications. Additionally, profile-related requirements can be enforced such as the country of residence and accuracy in previously solved tasks. In order to use Mechanical Turks' work force, a requester must create a new project that will wrap all the data regarding the task. A project or job represents a HIT template, containing common parameters (e.g. name, description, keywords, rewards per assignment, assignments per hit, allotted time per assignment) and the layout design in HTML. A project can be created from scratch or from a predefined project template such as the ones present in fig. 5.

Afterwards, a set of HITs can be submitted to Mechanical Turk using the created project. Along with this request, a CSV (Comma Separated Values) file must be uploaded. Each row of the CSV represents a HIT and contains the required data to present the HIT using the layout design defined in the project. Finally, the requester can approve (the reward is given) or disapprove (the reward is not given) the results of worker assignments.

Mechanical Turk does not include an aggregation mechanism. According to the project configuration, the assignment results given directly to the requester may contain multiple answers to each HIT.

As a worker, several assignments for the same HIT can be accepted. If a specific qualification is required for the HIT, the corresponding qualification test must be passed first.

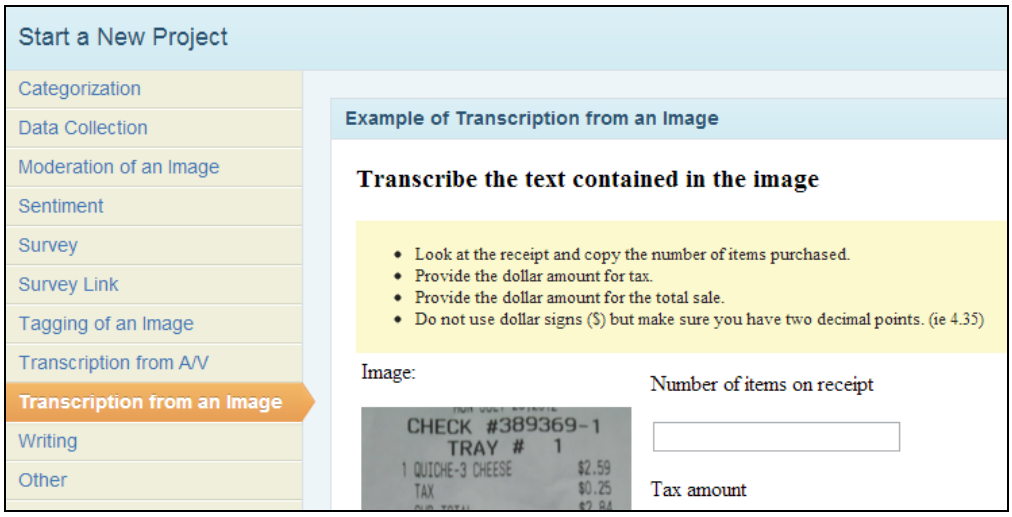


**Fig. 5 - Pre-defined project templates in Mechanical Turk.**

### *CrowdFlower*

CrowdFlower is a micro-task CS platform that distributes micro-tasks over several CS channels such as Mechanical Turk and Crowd Guru. Currently, more than fifty channels are supported.

A requester starts by submitting a job to CrowdFlower. A job is a template and an aggregation of multiple micro-tasks, known as units.

Several workers can work on the same unit. The result of this work for a single unit is called judgement. In this sense, one unit can have multiple judgements given by multiple workers (even one worker can give two or more judgements to the same unit).

Some of these units can be gold units. Gold units come with a reference judgement (correct answer) and are used to validate the work done by workers. If a worker gives a wrong judgement to a gold unit, their confidence degree will decrease until their judgements are not accounted for in the aggregated job results. Besides the specification of common parameters (e.g. keywords, judgements per worker, judgements per IP, allowed countries, judgements per unit), a CrowdFlower job contains the layout design defined in CML (CrowdFlower Markup Language). The CML is a language that provides an abstraction over HTML objects and allows interaction with the unit data (see fig. 6), which is also uploaded to CrowdFlower through a CSV file.

After supplying all the required data (units, gold units, CML form, and other configuration parameters) the requester can order judgements. In this order, one or more distribution channels can be selected (e.g. Mechanical Turk, Crowd Guru, SurveyHunt, Earn The Most).

When the job is finished, a full report is given. Additionally, aggregated, source, gold unit and worker reports are supplied in CSV format. The aggregation mechanism of CrowdFlower is based on a majority voting scheme that may exclude judgements according to the supplied gold units.

A worker participates in a job by giving judgements over sets of units presented on a single page. If allowed, a single worker can submit as many judgements as units. Taking into account the existence of gold units, this amount may even exceed the amount of units.

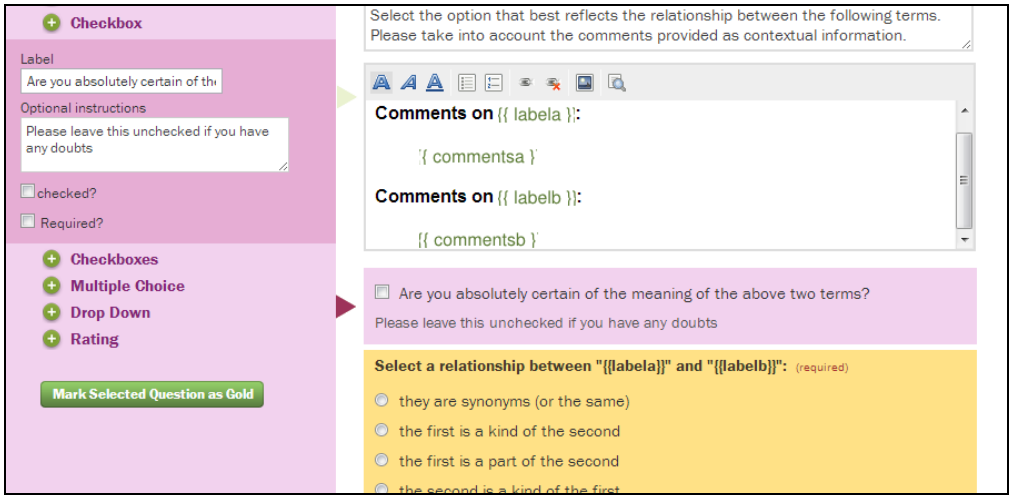


**Fig. 6 - CML graphical interface editor in CrowdFlower.**

### *ShortTask*

ShortTask is an online labour market similar to Mechanical Turk. Consequently, a requester can create a task template or job, which can be used to order multiple units. These tasks will then be assigned to workers.

Although the terms task and assignment are used, their application is not coherent. On some pages, task is equivalent to the term HIT; on others, it is equivalent to the term Project. Furthermore, even though there is a parameter for selecting the amount of assignments in the task template, requesting multiple answers for one task when ordering was not possible while experimenting with ShortTask.

### *MicroWorkers*

MicroWorkers[12] is a CS platform that focuses solely on the distribution of a micro-task amongst multiple workers. Unlike in Mechanical Turk and CrowdFlower, the process of building the job is significantly less structured and simpler. Instead of an interface for solving tasks structured with a markup language, the worker is given a set of instructions in natural language. These instructions are fixed for each job, meaning that submission will result in multiple workers performing the same task.

Given these limitations, jobs in MicroWorkers mostly involve work such as filling surveys, searching, rating, clicking, bookmarking, commenting, downloading, and installing applications. These are often associated with forums and websites such as Google, YouTube, Facebook and Twitter.

Probably due to the nature of its jobs, MicroWorkers does not implement any specific worker selection mechanism besides the possibility of targeting specific countries. Also, no automatic worker assessment is performed. Requesters may instead ask for the submission of proof regarding the task completion.

### *CloudCrowd*

CloudCrowd is a CS platform originally implemented as a Facebook application. Unlike the previously described platforms, users can only register as workers. Comparatively, CloudCrowd is more selective regarding workers and their expertise. In order to work on projects, workers need to get credentials (see fig. 7). These credentials are given after successfully solving specific tests with multiple levels of difficulty. Additionally, a credibility score is given to the worker. The credibility score value can increase or decrease according to the worker credentials and work feedback (e.g. incorrect answers).

In order to assess the credibility of workers, CloudCrowd uses check tasks. Check tasks are tasks with known answers, similar to gold units in CrowdFlower.

Workers can browse a list of projects and credentials, grouped by the following types: writing, editing, categorization, research, data entry, translation, and other. For each project, the required credibility and credentials are presented,

[12] https://microworkers.com

along with the monetary reward and availability. The availability represents the amount of tasks available to be solved in the project.

From the analysis of the CloudCrowd worker interface, it is unknown if the same task is given to multiple workers or if there is any aggregation mechanism available to requesters.

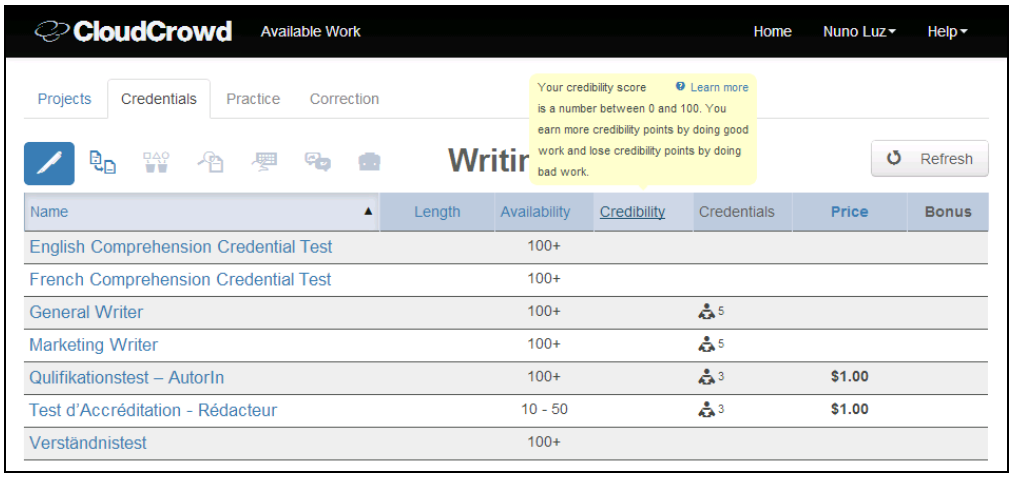


**Fig. 7 - Writing credential tests in CloudCrowd.**

### *TurKit*

Mechanical Turk focuses on independent tasks that can be executed in parallel. However, it does not support the creation of workflows of dependent tasks. TurKit is an API, built on top of Mechanical Turk, for running iterative tasks that provides an environment for the creation of workflows that connect multiple dependent tasks (Little et al. 2010).

Using the crash and rerun model, it provides an abstraction over the specificities and synchronization issues of Mechanical Turk, allowing the developer to focus on imperative ordinary function calls.

The crash and rerun model follows the premise that it is cheap to rerun an entire program up to the point where it crashed, as long as it runs locally. For remote and costly operations (e.g. HIT requests) the results must be stored in a database so that they will be accessible in future reruns.

The TurKit API is implemented in Java and allows the implementation of Mechanical Turk workflows in a JavaScript crash and rerun environment. This environment features a set of function directives for an easy and incremental implementation of scripts.

The TurKit provides a development environment that is available both as a stand-alone application and as a web application. The web application runs on Google App Engine.

### *Turkomatic*

Turkomatic follows a divide-and-conquer approach to plan work featuring micro-task workflows partially designed by workers. It works over Mechanical Turk and is defined as "a crowdsourcing interface that consults the crowd to design and execute workflows based on user requests" (Kulkarni et al. 2011).

Workers start by dividing the requested task into subtasks that will be solved by other workers. This process can be iterative, generating a tree of sub-tasks. The final results are later combined by workers into an adequate solution.

Turkomatic allows the requester to manage the generated workflow and its execution through a visual workflow editor.

### *CrowdForge*

CrowdForge is a general purpose framework for distributed processing that provides a scaffolding for complex human computation tasks (Kittur et al. 2011).The approach features a set of task coordination strategies that allow multi-level and dynamic partitioning of tasks, the specification of task workflows, quality control tasks and aggregation of results.

Finding inspiration in Google's Map Reduce framework, CrowdForge defines three types of subtasks: (i) partition tasks, (ii) map tasks and (iii) reduce tasks. Partition tasks divide a larger task into smaller subtasks. In map tasks, one or more workers process a task. Finally, in reduce tasks the processing results of multiple workers are merged into a single output.

The CrowdForge prototype presented in (Kittur et al. 2011) consists in a web interface allowing the design of complex tasks, along with a backend server that interacts with Mechanical Turk. The system manages a workflow of Mechanical Turk HIT templates, which can represent partitions maps or reduce tasks.

### *CrowdWeaver*

TurKit, CrowdForge and Jabberwocky provide an environment for the design and execution of complex CS tasks through structured languages and non-visual rep-

resentations. With the current trend on micro-task workflows, work on their management and visualization has started to emerge. Kittur et al. (2012) state that one of the major issues faced by employers working with crowds lies in the complexity of linking tasks and forming workflows. In this sense, they identify several challenges in visually managing CS workflows and present a system for visualization and management of complex tasks, entitled CrowdWeaver.

CrowdWeaver works on top of CrowdFlower and features the creation and monitoring of task workflows, the management and reuse of templates with human and machine tasks, the tracking and notification of crowd factors such as price and quality, and support for real-time experimentation. The interface provides a mental representation of the task workflow, which can be saved and reused in further instantiations of the workflow.

Several machine tasks, which mainly manipulate input and output, are supported. These include divide, concatenate, pair, and permute. Additionally, custom machine tasks are allowed.

### *Jabberwocky*

Similarly, Jabberwocky (Ahmad et al. 2011) also employs the MapReduce approach in a framework featuring a high-level abstraction task modelling language. It allows modelling complex tasks and workflows in which the advantages of multiple worker communities can be harnessed. These communities can be local or found in social networks and other CS systems. In the case of local communities, workers can be identified during the CS process. For social network communities, expertise data may be extracted from the social network APIs.

Jabberwocky is formed by three layers: the (i) base layer is called Dormouse, followed by the (ii) ManReduce layer, with the (iii) Dog layer on top. The Dormouse layer provides an abstraction over human (crowd workers) and machine computational units. Unlike other CS frameworks such as Mechanical Turk and CrowdFlower, each computational unit registered under Dormouse can be uniquely identified during workflow executions. Besides featuring its own worker community, Dormouse can crowdsource tasks to external CS platforms.

The ManReduce layer is a programming framework, written in Ruby, responsible for facilitating complex data processing tasks in Dormouse. It features map and reduce steps which can be computed by either humans or machines.

The top layer, called Dog, represents an abstraction scripting language for modelling tasks. Dog works over the low-level ManReduce framework and focuses on reusability, maintainability and ease-of-use.

### *Comparative Analysis*

While featuring the integration of a wide range of CS platforms, CrowdFlower also features the CML, an abstraction language for building task graphical user interfaces. Other analysed platforms only support direct HTML usage.

According to our analysis, Jabberwocky represents a highly complete CS platform compared to the others. It adds several features to CrowdForge, such as:

- Worker profiles and social networks;
- Abstraction over human and machine computational units;
- Multiple worker sources (e.g. social networks, external CS communities).

**Table 1 - Comparison of different CS systems.**

| System | Relies on | Complex Tasks | Task Methods | Worker Assessment | Aggregation |
|---|---|---|---|---|---|
| **MTurk** | Self | No | Task Templates | Qualification Tests | Manual |
| **CrowdFlower** | Several | No | Task Templates | Gold Units | Yes |
| **ShortTask** | Self | No | Task Templates | Manual | Manual |
| **MicroWorkers** | Self | No | Task Templates | Manual | N/A |
| **CloudCrowd** | Self | - | - | Credential Tests and Credibility | - |
| **CrowdForge** | MTurk | Workflows | Map Reduce | (MTurks') | Yes |
| **Jabberwocky** | Self/Several | Workflows | Map Reduce | User Profiles | Yes |
| **Turkomatic** | MTurk | Workflows | Divide and Conquer | (MTurks') | Yes (Workers) |
| **Turkit** | MTurk | Workflows | Crash and Rerun | (MTurks') | Yes (Workers) |

**Table 2 - Different terminologies employed by different CS systems.**

| Meaning | MTurk | CrowdFlower | ShortTask | CloudCrowd | MWorkers |
|---|---|---|---|---|---|
| **Wrapper of CS data and tasks** | Project | Job | Task Template | Project | Campaign or Job |
| **Unit of the task to be solved** | HIT | Unit | Task | Task | Task |
| **Assignment of a unit to a worker** | Assignment | - | Assignment | - | - |
| **Answer given to an assignment** | Answer | Judgement | - | Answer | - |
| **Ref. unit with correct answer** | N/A | Gold Unit | N/A | Check Task | N/A |
| **The employee** | Worker | Worker | Solver | Worker | Worker |
| **The employer** | Requester | Requester | Seeker | - | Employer |
| **Worker expertise requirement** | Qualification | N/A | N/A | Credentials | N/A |

CloudCrowd has a rigorous and complete assessment system compared to most CS platforms, incorporating the benefits of both Mechanical Turks' and CrowdFlowers' assessment systems.

Table 1 contains an overview and comparison of the analysed CS systems. The "relies on" column indicates if the system is able to relay the CS process to other systems. The "complex tasks" column describes, if available, how complex tasks are formed and handled. The third column, "task methodologies", enumerates different methodologies employed by the system in designing, maintaining, and executing tasks. The "worker assessment" column refers to how the CS system evaluates the quality of the work performed by workers. Finally, the "aggre-

gation" column describes, if available and applicable, automatic result aggregation procedures that merge the work of several workers.

The terminology employed in the CS domain often varies from platform to platform. Table 2 wraps the terminology used in four different CS platforms.

The first concept found in table 2 represents a wrapper of the whole CS process. It contains all the data required for the execution of the task or task workflow, including inputs and outputs. A task (e.g. tag an image) can have several units of work (e.g. the actual images to be tagged). This concept is represented in the second row of table 2. The following concept depicts assignments of specific units to workers (the ones who solve the task). For each assignment, the worker must submit an answer (fourth row concept).

In some cases, for quality control purposes, units are submitted by the requester with an already known correct answer. This kind of unit is present in the fifth row of table 2. Furthermore, the requester (the one who requires a certain task to be solved) can specify expertise requirements that workers must satisfy in order to be electable for solving the task. The terminology for this concept can be found in the last row of table 2.

## Ontology of Task-Oriented Systems

Although classifications of CS systems already exist (Doan et al. 2011; Quinn and Bederson 2011; Yuen et al. 2011a), they often focus on dimensions that are domain-specific or of particular interest from a business standpoint. In this sense, a new systematization of these classifications is provided, integrating an analysis of the previously described systems.

The presented classification, in table 3, focuses on technical and domain-independent dimensions of systems that fit under both the CS and HC definitions. It is built as a poly-hierarchy of categories, meaning that a CS system can fit under a number of these categories. In this sense, the classification is presented as an ontology. Although not considered in the context of this work, an artefact of this ontology can be built using, for instance, the OWL (Web Ontology Language).

As a reference for the classification, the following dimensions for task-oriented systems were identified:

- Nature of collaboration (Doan et al. 2011);
- Architecture (Doan et al. 2011);

- Worker selection;
- Worker assessment and quality control;
- Worker motivation;
- Task creation and configuration;
- Task management;
- Task execution;
- Task result aggregation.

### Nature of Collaboration

The explicit and implicit categories present in the "nature of collaboration" dimension are introduced by Doan et al. (2011). In explicit CS systems the users are aware that they are working towards solving specific tasks. Some examples of explicit CS systems include Mechanical Turk, CrowdFlower and ShortTask.

Implicit CS systems usually retrieve work behind a system with a different purpose. These include games such as the ESP Game (Von Ahn 2009), Tag-a-Tune (Quinn and Bederson 2011), and Foldit (Cooper et al. 2010). ReCAPTCHA (Little et al. 2010) is also an example of an implicit CS system where users work as book and document digitizers by filling CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) forms.

In this classification, it is intended that the implicit and explicit categories are disjoint. Thus, it is not possible for one CS system to be classified as both implicit and explicit. Since CS games fit under the implicit category, a game subcategory has been included in the classification.

### Architecture

The architecture dimension encompasses the standalone and piggyback categories, which are also introduced by Doan et al. (2011). A standalone CS system is independent of other CS systems. It has its own worker community and does not require the distribution of tasks among other systems (e.g. MTurk).

A piggyback CS system relies on external worker communities. They usually focus on the process of building the task and distributing work between other CS systems. One of the best examples of such a system is CrowdFlower.

The given definition for these two categories is not disjoint. In this sense, a CS system, such as Jabberwocky, fits both categories.

**Table 3 - Classification of CS and HC systems according to different dimensions.**

| Dimension | Category | Sub-category |
|---|---|---|
| **Nature of Collaboration** | Explicit | |
| | Implicit | Game |
| **Architecture** | Standalone | |
| | Piggyback | |
| **Worker Selection** | Community Scope | Multiple Community |
| | | Single Community |
| | Worker Scope | Assessment Filtering |
| | | Expertise test Filtering |
| | | Profile (social) Filtering |
| | | Demographic Filtering |
| **Worker Assessment and Quality Control** | Manual | |
| | Automatic | Reference Units |
| | | Assessment Tasks |
| **Worker Motivation** | Monetary | |
| | Enjoyment | |
| | Reputation | |
| **Task Creation and Configuration** | Instantiation | Manual |
| | | Templates |
| | UI Construction | Plain Text |
| | | HTML |
| | | Proprietary Language |
| | Data Structure | Plain Text |
| | | CSV, JSON (No Schema) |
| **Task Management** | Non-Visual | |
| | Visual | Construction |
| | | Monitoring |
| **Task Execution** | Simple | |
| | Complex (Workflows) | Crash and Rerun |
| | | Divide and Conquer |
| | | Partition-Map-Reduce |
| **Task Result Aggregation** | Manual | |
| | Automatic | Voting Scheme |
| | | Assessment-based |
| | | Crowdsourced |

## Worker Selection

When a task is published, it becomes available to a certain amount of workers. These workers usually belong to at least one community, and may be filtered according to different characteristics such as their profile information, demographics (e.g. country of residence), or expertise. In this sense, six types of worker selection strategies that act in two different scopes (the community scope, and the worker scope) were identified (see table 4).

**Table 4 - Classification of CS systems according to the worker selection dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Community Scope** | Multiple Communities | Jabberwocky |
| | Single Community | MTurk, ShortTask |
| **Worker Scope** | Assessment Filtering | MTurk, CloudCrowd |
| | Expertise Test Filtering | MTurk, CloudCrowd |
| | Profile (Social) Filtering | |
| | Demographic Filtering | CrowdFlower |

The community scope encompasses CS systems that either rely on multiple communities of workers or on a single worker community. These two categories are disjoint.

In the worker scope there are four categories: assessment filtering, expertise test filtering, demographic filtering, and profile (social) filtering. In assessment-based filtering, the task is distributed between workers that satisfy requirements regarding reputation and performance in previously solved tasks. Expertise test filtering consists of providing test tasks that assess worker expertise in a specific domain of knowledge, or in performing a specific type of task. The measurement of expertise allows the enforcement of a minimum expertise value for the worker to be able to participate.

Demographic filtering selects workers that satisfy certain statistical characteristics of attributes such as age and country of residence. Profile (social) filtering represents a superset of demographic filtering. It enforces requirements regarding the worker personal and social profile, which may include demographic attributes. One example is the enforcement of a minimum or maximum value for social worker relationship closeness (Wasserman and Faust 1994).

### Worker Assessment and Quality Control

Assessing the performance of workers in specific tasks or in general is an important dimension of CS. Worker assessment can be employed in different phases and in other dimensions of the CS process such as in worker selection and task result aggregation, often having a significant impact on the quality of the final task results.

According to the current state of the art, assessment strategies are either manual (performed by the requester) or automatic (performed by the CS system).

**Table 5 - Classification of CS systems according to the worker assessment and quality control dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Manual** | | ShortTask, MicroWorkers |
| **Automatic** | Reference Units | CrowdFlower |
| | Assessment Tasks | Turkomatic, CrowdForge |

As presented in table 5, two automatic assessment strategies were identified: reference units, and assessment tasks. An assessment through reference units consists of using test tasks with known answers to evaluate worker performance. Another strategy is to crowdsource the assessment of the answers to other workers through an assessment task.

### Worker Motivation

For users to participate and remain as workers, they require some kind of motivation, usually given by the CS system or the requester.

Most HC and CS systems provide small monetary rewards. Others, however, use different means of motivating workers. While in CS games the motivational aspect is usually recreation and enjoyment, reputation has also been proven a viable source of motivation (Porter 2008).

In this sense, three worker motivation strategies are identified as the following non-disjoint categories:

- Monetary (e.g. MTurk and CrowdFlower);
- Enjoyment (e.g. Tag-a-Tune and the ESP Game);
- Reputation (not so widely exploited since workers are often anonymous).

## Task Creation and Configuration

The task creation and configuration dimension encloses all the special characteristics of the task creation and configuration process. So far, it mainly consists of the definition of the User Interface (UI), upload of data, and configuration of the task request through parameters such as monetary reward per unit and target worker residence countries.

Referring to this dimension, a classification is presented in table 6 that takes into account how the task instance is created, how the construction of the worker UI is performed, and which data structures are consumed by the CS system. None of the included categories and sub-categories are disjoint.

Most CS systems allow both the manual and template-based instantiation of the task. Also, the CSV format is commonly used in CS systems to allow requesters to upload task data.

**Table 6 - Classification of CS systems according to the task creation and configuration dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Instantiation** | Templates | MTurk, ShortTask |
| | Manual | CrowdFlower, MTurk |
| **UI Construction** | Proprietary Language | CrowdFlower |
| | HTML | MTurk |
| | Plain Text | MicroWorkers |
| **Data Structure** | CSV, JSON (No Schema) | CrowdFlower, MTurk |
| | Plain Text | MicroWorkers |

## Task Management

So far, the management of tasks has been performed through simple, mostly textual, web UI. With the advent of several complex task CS systems, however, a need for visually building and managing task workflows has emerged.

Turkomatic is one of the first CS systems introducing task visualization by presenting a complex task construction environment where the requester is able to visually build a workflow of tasks. More recently, CrowdWeaver takes complex

task visualization a step further by providing an environment not only for visually building workflows of tasks but also for monitoring their progress.

Taking into account the current state of the art, four categories for the task management dimension are presented in table 7.

**Table 7 - Classification of CS systems according to the task management dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Non-Visual** | | CrowdFlower, MTurk |
| **Visual** | Construction | CrowdWeaver, Turkomatic |
| | Monitoring | CrowdWeaver |

In this dimension, only the non-visual and visual categories are disjoint. The construction category refers to CS systems that provide a visual UI for building tasks. The monitoring category, on the other hand, refers to CS systems that allow the visual monitoring of the task progress in real time.

## Task Execution

The task execution dimension refers to how the task is structured and processed by the CS system. Under this dimension, a task can either be simple or complex (a workflow of simple tasks). For complex tasks, several strategies are employed. These are present in the classification with the sub-categories shown in table 8.

**Table 8 - Classification of CS systems according to the task execution dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Simple** | | CrowdFlower, MTurk |
| **Complex** | Crash and Rerun | Turkit |
| | Divide and Conquer | Turkomatic |
| | Partition-Map-Reduce | CrowdForge, Jabberwocky |

The crash and rerun category includes systems that allow modifying and restarting the execution of the workflow without additional CS costs. Divide and conquer approaches try to split the task into smaller units before requesting a final solution. Partition-map-reduce are a specific case of divide and conquer approaches that include a result aggregation phase.

### Task Result Aggregation

Often, the same task is solved by multiple workers, leading to several solutions for the same task. Despite providing redundancy, it requires the aggregation and processing of the resulting data in order to reach a final and unique solution.

The aggregation of results can be either manual (performed by the requester of the task) or performed automatically using an aggregation strategy. Commonly used aggregation strategies include the application of voting schemes and filtering through worker assessment. CrowdFlower employs both of these aggregation strategies. It uses a majority voting scheme to select results that are most likely to be correct, excluding workers with poor assessment values (those that gave wrong answers to check tasks).

The results of the algorithms employed during worker selection can also provide important data for result aggregation. Both CloudCrowd and Mechanical Turk implement mechanisms to assess expertise (qualifications or credentials) in order to filter workers. Giving more prominence to the answers of workers with higher degrees of expertise could lead to better results.

Another approach is to crowdsource the result aggregation as an aggregation task. This strategy is employed in systems that use the partition-map-reduce task execution strategy.

Table 9 presents the categories identified for the task result aggregation dimension.

**Table 9 - Classification of CS systems according to the task result aggregation dimension.**

| Category | Sub-category | Examples |
|---|---|---|
| **Manual** | | ShortTask |
| **Automatic** | Voting Scheme | CrowdFlower |
| | Assessment-based | CrowdFlower, CloudCrowd |
| | Crowdsourced | CrowdForge, Jabberwocky |

## Challenges

Since the appearance of the first CS systems for micro-tasks, many more have emerged. Their continuous use has led to several experiments and studies that often focus on user motivation and quality control (Yuen et al. 2011b).

User motivation has been addressed in several ways, from providing enjoyment and relying on altruism, to giving monetary rewards (Faridani et al. 2011). The latter has been the subject of some criticism, due to the creation of cheap labour marketplaces (Harris 2011). In the specific case of task CS systems, the current monetary rewards are not sufficient to be a primary source of income, and often they are not enough to serve as a motivator (Mason and Watts 2010; Paolacci et al. 2010). In these cases, workers actually participate out of altruism, curiosity, or simply to keep themselves busy. Still, motivating and retaining workers over time remains a challenge for any CS system.

Quality control has been studied in different application domains and addressed differently by a variety of CS systems. In general, assessment methods are employed either before (during the worker selection step) or after (during the worker assessment step) the participation of the worker. Among the current assessment strategies are expertise tests in certain domains, asking questions for which the answer is known and analysing performance in previous tasks. Although several quality control strategies already exist, the advent of CS of complex tasks has brought new challenges to different dimensions of the CS process, including specification, flow control, quality control, and visualization. In the specific case of quality control, and due to the presence of a micro-task workflow structure, a deviation or error in one task can accumulate with those of the following tasks.

Besides quality control, the flow control assumes special relevance. In fact, the CS of complex tasks establishes requirements regarding worker selection that are often discarded in simple micro-task CS systems. One of these requirements is worker identity. The fact that most CS systems regard the micro-task as the top-level unit of work, leads to loss of identity information from one micro-task to the other. This can easily become a challenge when the worker, as an identifiable individual, is required to participate in different steps of the task workflow.

Another challenge when dealing with complex tasks is the aggregation and visualization of results. At some point, since multiple micro-tasks are involved in a workflow, tracing a specific result and asserting the causes and facts that led to it becomes a necessity. Currently, both the results presented by most CS systems and the input requested from workers are poorly structured and, in some cases, in

natural language. Since the output of a micro-task might be the input given to the execution of a computer algorithm micro-task, both the specifications (structure) of the task and the task domain of knowledge must be understandable and interpretable by both humans and machines. Furthermore, there is a necessity for a specification that is able to capture iterative steps formed by complex flow conditions.

Enforcing structured machine-readable data also facilitates the implementation of strategies for tracing workflow results. Still, special care is needed so that such a structure does not increase the complexity and cumbersome nature of solving tasks from the worker's perspective.

Overall, the following main challenges were identified:

- Specification of complex tasks, namely:
    - Iterative, interactive and incremental tasks;
    - Support for complex flow control conditions, both internal and external;
    - Structuring the task specification, including its domain knowledge, in a way understandable and interpretable by both workers and machines;
- Specification of the human-machine interaction, namely:
    - Specifying worker selection restrictions for different phases of the workflow;
    - Identifying workers across the workflow in order to permit iterative workflows;
    - Tracing results and obtaining and providing justifications;
- Control quality in complex task workflows:
    - Assessing and reducing the impact of low quality answers across the workflow;
    - Using (social) profile information to assess worker's expertise and relevance, and to enhance context;
- Visualization and reporting.

## Possible Directions

It is curious to notice that while multi-agent systems usually strive to create computer entities that mimic human behaviour, HC and CS strives for the extraction

of computational power from humans. So far, this has been done by creating systems that structure the task so humans can execute it, in similar ways to machines. In a sense, the one-way computer-human bridge built over years and years of research on multi-agent systems might as well present the foundations required for building another bridge in the opposite direction.

In particular, a structured and semantically-enriched representation of the micro-task workflows and their data is required to allow a better integration of human computation and machine computation efforts. Ontologies as a formal representation mechanism are currently "the best answer to the demand for intelligent systems that operate closer to the human conceptual level" (Obrst et al. 2003).

While providing structure and semantics, ontologies can also leverage:

- Template re-use and extensibility;
- Semi-automatic generation of worker interfaces;
- Contextual information in complex workflows (tackling flow and quality control);
- Justifications for results (tackling result traceability in flow control);
- Semantics for further automatic processing of data (tackling complex task specification of the task and the task domain).

Detailed representations (ontologies) of micro-tasks and their dependencies through the specification of their domain (input, output and context) can be analysed to present relevant contextual information to workers and detailed structured reports to requesters (Luz et al. 2014). Although this is not particularly interesting in single task CS systems, it is relevant when dealing with complex task workflows and their input/output dependencies.

Another dimension of CS micro-task systems in which further research can potentially lead to new CS solutions is worker selection and, in particular, profile (social) filtering. The application of worker filters according to social network connections and data helps to tackle some of the previously identified quality control challenges, and may lead to interesting results in specific types of tasks where social network properties such as homophily (the tendency to bond with similar others) and centrality (Wasserman and Faust 1994) usually play an important role. This is the case for personal or private tasks such as selecting a good local restau-

rant or identifying acquaintances in old family photographs (a similar process to tagging photographs, which is often found in social networks).

Micro-task CS is a research field with a wide variety of application domains. Also, given the complexity of the dimensions involved in the CS process, there is room for many different approaches. In this sense, the proposed ideas and possible directions are but a fraction of the future research and development possibilities.

## Conclusions

Over the last decade, HC and CS have been merged to create multiple systems for problem solving on a wide scale. These systems are starting to facilitate the man-computer symbiosis envisioned by Licklider (1960) in the 1960s.

The increasing popularity of CS has led to a variety of commercial and non-commercial applications in different domains. Among these applications, a small set of CS platforms oriented towards problem (task) solving has emerged. These platforms regard the problem (or job) as sets of micro-tasks that can be solved redundantly in one step by multiple users. In complex jobs where a workflow of micro-tasks is required, new challenges emerge. This new trend in CS has led to some interesting workflow oriented systems such as CrowdForge and Jabberwocky.

Overall, task-oriented CS systems share many commonalities. The presented systematization of the current state of the art in task-oriented CS systems, which includes a classification of systems, aids the quick identification of these commonalities and provides an overview of the current challenges and possible research directions.

Misunderstanding is an issue that might often emerge due to the use of different terminologies by task-oriented CS systems. In this sense, a mapping of the different terminologies employed in CS is provided, along with the relationships between each term.

Although several task-oriented CS systems have emerged in a short period of time, the employed approaches are often the same. In fact, a generalization of the CS process (for simple and complex tasks) using a common/mapped terminology is proposed.

Finally, a focus is given to the identification and systematization of challenges and possible approaches, especially concerning complex tasks. This shows that there is still a great deal of room for further research and development.

## Acknowledgements

This work is part-funded by ERDF – European Regional Development Fund through the COMPETE Programme (Operational Programme for Competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within the PhD grant SFRH/BD/70302/2010 and by the projects AAL4ALL (QREN11495), World Search (QREN 13852) and FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012). The authors also thank Jane Boardman for her assistance proof reading the document.

## References

Ahmad S, Battle A, Malkani Z, Kamvar S (2011) The jabberwocky programming environment for structured social computing. Proc. 24th Annu. ACM Symp. User Interface Softw. Technol. pp 53–64

Von Ahn L (2009) Human computation. 46th ACMIEEE Des. Autom. Conf. pp 418–419

Brabham DC (2008a) Crowdsourcing as a model for problem solving an introduction and cases. Converg Int J Res New Media Technol 14:75–90.

Brabham DC (2008b) Moving the crowd at iStockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application. First Monday 13:1–22.

Chklovski T (2003) Learner: a system for acquiring commonsense knowledge by analogy. Proc. 2nd Int. Conf. Knowl. Capture. pp 4–12

Cooper S, Khatib F, Treuille A, et al. (2010) Predicting protein structures with a multiplayer online game. Nature 466:756–760. doi: 10.1038/nature09304

Doan A, Ramakrishnan R, Halevy AY (2011) Crowdsourcing systems on the world-wide web. Commun ACM 54:86–96.

Faridani S, Hartmann B, Ipeirotis PG (2011) What's the right price? pricing tasks for finishing on time. Proc. AAAI Workshop Hum. Comput.

Goodchild MF, Glennon JA (2010) Crowdsourcing geographic information for disaster response: a research frontier. Int J Digit Earth 3:231–241.

Gruber T (2008) Collective knowledge systems: Where the social web meets the semantic web. Web Semant Sci Serv Agents World Wide Web 6:4–13.

Harris CG (2011) Dirty Deeds Done Dirt Cheap: A Darker Side to Crowdsourcing. IEEE Int. Conf. Priv. Secur. Risk Trust. pp 1314–1317

Howe J (2006) The rise of crowdsourcing. Wired Mag 14:1–4.

Howe J (2008) Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business. Century

Ipeirotis PG, Provost F, Wang J (2010) Quality management on amazon mechanical turk. Proc. ACM SIGKDD Workshop Hum. Comput. pp 64–67

Kittur A, Chi EH, Suh B (2008) Crowdsourcing user studies with Mechanical Turk. Proc. SIGCHI Conf. Hum. Factors Comput. Syst. pp 453–456

Kittur A, Khamkar S, André P, Kraut R (2012) CrowdWeaver: visually managing complex crowd work. Proc. ACM 2012 Conf. Comput. Support. Coop. Work. pp 1033–1036

Kittur A, Smus B, Khamkar S, Kraut RE (2011) Crowdforge: Crowdsourcing complex work. Proc. 24th Annu. ACM Symp. User Interface Softw. Technol. pp 43–52

Konstas I, Stathopoulos V, Jose JM (2009) On social networks and collaborative recommendation. Proc. 32nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. Boston, MA, USA, pp 195–202

Kulkarni AP, Can M, Hartmann B (2011) Turkomatic: automatic recursive task and workflow design for mechanical turk. Proc. 2011 Annu. Conf. Ext. Abstr. Hum. Factors Comput. Syst. pp 2053–2058

Levine SS, Kurzban R (2006) Explaining clustering in social networks: towards an evolutionary theory of cascading benefits. Manag Decis Econ 27:173–187. doi: 10.1002/mde.1291

Licklider JCR (1960) Man-computer symbiosis. IRE Trans Hum Factors Electron 4–11.

Little G, Chilton LB, Goldman M, Miller RC (2010) Turkit: human computation algorithms on mechanical turk. Proc. 23nd Annu. ACM Symp. User Interface Softw. Technol. pp 57–66

Luo S, Xia H, Yoshida T, Wang Z (2009) Toward collective intelligence of online communities: A primitive conceptual model. J Syst Sci Syst Eng 18:203–221.

Luz N, Silva N, Maio P, Novais P (2012) Ontology Alignment through Argumentation. 2012 AAAI Spring Symp. Ser.

Luz N, Silva N, Novais P (2014) Generating Human-Computer Micro-task Workflows from Domain Ontologies. In: Kurosu M (ed) Hum.-Comput. Interact. Theor. Methods Tools. Springer International Publishing, pp 98–109

Ma H, Zhou D, Liu C, et al. (2011) Recommender systems with social regularization. Proc. Fourth ACM Int. Conf. Web Search Data Min. pp 287–296

Mason W, Watts DJ (2010) Financial incentives and the performance of crowds. ACM SigKDD Explor Newsl 11:100–108.

Obrst L, Liu H, Wray R (2003) Ontologies for corporate web applications. AI Mag 24:49.

Paolacci G, Chandler J, Ipeirotis P (2010) Running experiments on amazon mechanical turk. Judgm Decis Mak 5:411–419.

Porter J (2008) Designing for the Social Web. Peachpit Press

Quinn AJ, Bederson BB (2011) Human computation: a survey and taxonomy of a growing field. Proc. 2011 Annu. Conf. Hum. Factors Comput. Syst. pp 1403–1412

Sarasua C, Simperl E, Noy NF (2012) CrowdMap: crowdsourcing ontology alignment with microtasks. Springer, pp 525–541

Singh P, Lin T, Mueller ET, et al. (2002) Open Mind Common Sense: Knowledge acquisition from the general public. Move Meaningful Internet Syst. 2002 CoopIS DOA ODBASE. Springer, pp 1223–1237

Surowiecki J (2004) The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations. N. Y. Doubleday

Wasserman S, Faust K (1994) Social Network Analysis: Methods and Applications, 1st ed. Cambridge University Press

Willett W, Heer J, Agrawala M (2012) Strategies for crowdsourcing social data analysis. Proc. 2012 ACM Annu. Conf. Hum. Factors Comput. Syst. pp 227–236

Yuen M-C, King I, Leung K-S (2011a) A Survey of Crowdsourcing Systems. IEEE Int. Conf. Priv. Secur. Risk Trust

Yuen M-C, King I, Leung K-S (2011b) Task matching in crowdsourcing. Internet Things IThingsCPSCom 2011 Int. Conf. 4th Int. Conf. Cyber Phys. Soc. Comput. pp 409–412