Supporting Complex Work in Crowdsourcing Platforms: A View from Service-Oriented Computing

Lu Xiao
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia,
Email: luxiao@cse.unsw.edu.au

Hye-Young Paik
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW 2052, Australia,
Email: hpaik@cse.unsw.edu.au

***Abstract*—Crowdsourcing is changing the way people work and solve problems from "in-house working" to "public outsourcing". Most online crowdsourcing platforms perform two main functions: (i) allowing users to advertise their tasks and (ii) helping them find candidate workers. However, they do not support crowdsourcing of complex work consisting of interdependent tasks. Those tasks require not only having simple task/worker pairs, but also coordinating multiple workers together for completion of the crowd work. In this paper, we propose a conceptual framework to bring the coordination support into current online crowdsourcing platforms. In our framework, each crowd worker is modeled as a service that can be self-described, dynamically discovered and assembled into the complex crowd work; meanwhile, we define a workflow-based schema to structure and represent the complex crowd work. Then the crowdsourcing performance is initiated and managed by our coordination protocol.**

I. Introduction

The notion of crowdsourcing has been described as a distributed problem-solving model and production model for both individuals and organisations [1], [2]. It allows users to outsource their tasks or problems to the crowd on a community or even global scale. By utilising the collective intelligence on demand, it can solve those problems that need much of human computation. Identifying a person or location from a picture, solving a puzzle, voting for a design, etc., are typical examples.

Many online crowdsourcing platforms, such as Amazon Mechanism Turk [1], act as brokers between *problem or task requesters* and distributed *crowd workers*. They allow *requesters* (either individuals or groups) to advertise tasks with financial compensation, and help *crowd workers* find jobs. Currently, most of these advertised tasks are independent and small-scale pieces of work. It is insufficient for those platforms to support the crowdsourcing of creative, complex, and structured work that needs various professionals with diverse expertise and coordination of these highly skilled individuals [3]. A software developing or testing process, producing an academic paper or book, can exemplify these sophisticated and professional crowd work.

Service-Oriented Computing (SOC) has been commonly referred to as a software development paradigm to build complex and scalable systems in a distributed environment [4]. It encapsulates the functionality of each different software as a service, in which also specifies a flexible interaction via binding. This paradigm regards the heterogeneous applications on the web as the autonomous and self-described services that can be loosely coupled and assembled. Therefore, we can create a complex, distributed system by composing these services on a web scale instead of building it from scratch. We argue that a crowdsourcing platform can be seen as a large distributed system, in which each problem requester can be viewed as a *service requester* and each crowd worker can be treated as an autonomous *service provider*. In this sense, we could build a crowdsourcing platform that is capable of dealing with the complex crowd work mentioned above by discovering, composing, and coordinating multiple crowd workers.

A. Motivation

Here we describe a motivating scenario to highlight the challenges of crowdsourcing the complex work. In this scenario, a software development team wants to outsource their testing process to the crowd, due to the lack of certain resources (e.g. professional testers). As we can see in Fig. 1, this crowd work is structured and consists of several interdependent, professional tasks. Different tasks require different professionals. For example, to make test cases, the worker should be capable of analysing and understanding the business value or objective behind the software product, and transforming them into the formal description of testing scenarios; similarly, to perform either functional or non-functional testing, the worker should be capable of using some testing tools or writing some auto-testing scripts. Therefore, this crowd work is not one simple task/worker pair; instead, it needs to get multiple professionals involved and coordinate their performance, e.g. one worker cannot start his/her testing work until (s)he gets the test case specification from another. By crowdsourcing this testing process, the requester expects a platform to be capable of:

- defining the structured crowd work other than a simple task with expected output;
- discovering the appropriate workers from the crowd, in terms of their qualification, reputation, and motivation;
- coordinating crowd workers based on the dependencies between their work, e.g. the control-flow or data-flow dependency;
- supervising or monitoring the performance of crowd workers, and controling the quality of their work.

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

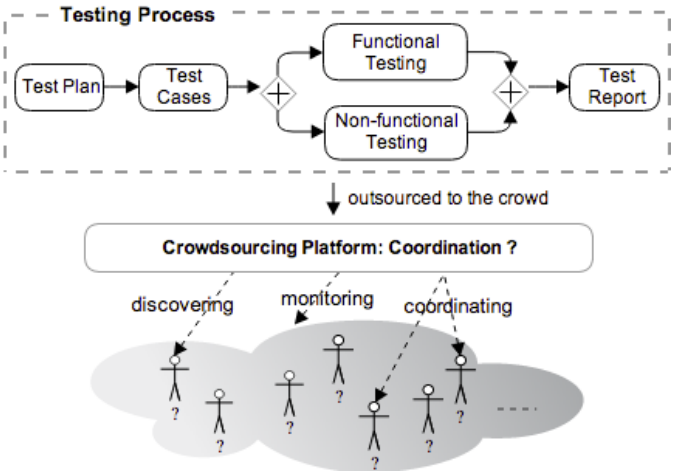

Fig. 1. An example of crowdsourcing complex work

We refer to the above as the *coordination requirement* for the crowdsourcing platform. To our best knowledge, most of current crowdsourcing platforms cannot meet this requirement comprehensively. Many popular ones, e.g. Mechanical Turk, Mobile-Works [2], Manpower [3], etc., mainly focus on matching or pairing a specific task with a particular crowd worker, since they target themselves as an online marketplace for job advertisers and seekers. They lack a systematic support of organisational behaviour, i.e. task decomposition, worker allocation, performance supervision, and quality assurance.

In this paper, we propose a workflow-based, service-oriented framework to better support the crowdsourcing of complex work through: i) a service-oriented profile that represents each crowd worker as a service, which can be described, discovered, and composed into the complex crowd work; ii) a workflow-based schema that defines the structured crowd work as a skeleton for the crowdsourcing management; iii) a coordination protocol that utilises the aforementioned schema and profile to initiate and manage the crowdsourcing process.

II. Related Work

Some professional crowdsourcing platforms, such as oDesk [4] and Freelancer [5], allow requesters to crowdsource professional jobs, e.g. website design, article writing, etc. Yet, they still focus on pairing one task with one worker without the coordination support. Besides commercial crowdsourcing platforms, some research work has been done to consider crowdsourcing complex work. [5] has presented a crowdsourcing framework named CrowdForge to support coordination of complex work in micro-task markets (i.e. MTurk). They propose a MapReduce-based approach to break down a complex task into a set of micro subtasks that can be solved by crowd workers. Then they manage the task flow and combine the output of micro-tasks as a final result for the complex crowd work. Yet, they mainly focus on task decomposition and recombination without the support of crowd worker allocation and supervision. [6] has presented a crowdsourcing tool named Turkomatic for crowd workers to collaboratively design workflows with the requester. Similar to [5], they mainly focus on task decomposition without allocation and supervision concern. We share the same vision of the future crowd work with [3]. In this vision, a crowdsourcing framework should be capable of supporting more complex and valued work by decomposing tasks, interpreting task dependencies and structure, finding and allocating appropriate crowd workers to perform those tasks, monitoring or supervising their performance, and evaluating their performance for high quality assurance.

[2]http://www.mobile-works.org/
[3]https://www.manpower.com.au/
[4]https://www.odesk.com/
[5]http://www.freelancer.com.au/

III. Concepts Overview

In this section, we briefly introduce the following key concepts in our approach.

- **Provider**: we represent each different, distributed crowd worker as a service provider using a service description profile. Through various services in this profile, each provider allows the requester to access the service information and reuse their capabilities for the outsourced work. More importantly, by abstracting each provider in the similar interface, we could draw an analogy from SOC to meet the coordination requirement above through service discovery and composition.
- **Crowd Work**: we define the crowd work outsourced by the requester in a workflow-based schema. This schema consists of a set of interdependent tasks whose information and structure provide a skeleton for the crowdsourcing platform to bind appropriate providers, coordinate them, request their services to produce data for the crowd work.
- **Coordination Protocol**: Based on the above provider and crowd work, we design a coordination protocol to initiate and manage the crowdsourcing process. Through the provider finding and binding, task execution and provider orchestration in this protocol, we can realise the coordination requirement in Sec. I.

In the following, we elaborate on each of them.

IV. Provider and a Service Description Profile

In our work, a crowd provider is depicted in his/her profile consisting of not only a set of personal details, but more importantly, a set of personal services that (s)he exposes to the public. One example can be seen in Fig. 2. Specifically, we define four different categories of personal services as below.

- **Capabilities Services**: this category is for requesters to reuse providers'capabilities for their outsourced tasks. More precisely, when a crowd provider finishes an outsourced task that (s)he has never done before, this high-level human task is translated into the low-level service definition in a service description language (i.e. WSDL or WADL), e.g. the input of this task can be translated into the service request parameters, the output becomes the response in WADL. Then the new defined service is registered in this provider's profile for others to request in the future for the similar tasks.

```
<Participant id="...">
    <name>Shawn</name>
    <gender>Male</gender>
    ...
    <!-- capabilities services -->
    <link title="automateTest"
        href="http://.../automate_testing" />
    ...
    <!-- information services -->
    <link title="getTitle"
        href="http://.../title_getter" />
    ...
    <!-- track record services -->
    <link title="calledServices"
        href="http://.../services_history" />
    <!-- condition services -->
    <link title="pay"
        href="http://.../services_cost" />
    <link title="availability"
        href="http://.../services_availability"
    <link title="contact"
        href="http://.../contact_method" />
</Participant>
```

Fig. 2. A provider profile example

- **Information Services**: crowd providers can not only provide services for requesters to reuse their skills or capabilities to handle the outsourced tasks, but also can provide services to access their personal information, e.g. name, title, position, domain, etc. These information in conjunction with capabilities services can be used to measure the qualification of a crowd provider.
- **Track Record Services**: this type is used to measure the quality of one's personal services and the reputation of this person by allowing requesters to view a historic record of their provided services. For instance, the number of capabilities services requested by others in the past; for each requested service, the requested times, rates, comments, and recommendations given by others, etc.
- **Condition Services**: this group is for one provider to declare the conditions and motivation of getting him/her involved in the crowd work, e.g. the available work period, contact method, and the pay.

Through the above services, we could find the appropriate providers and allocate them into the appropriate outsourced tasks in terms of their qualification, reputation, and motivation. Further, by representing each different, distributed provider in the similar profile, we could draw an analogy from the traditional SOC field to compose and coordinate them together to complete the complex crowd work. Also, by allowing individuals to manage their profiles, e.g. update or remove a personal service, it provides a way to capture the dynamically changing capabilities and information of providers as their skills and information evolve.

V. Crowd Work as a Workflow-based Schema

We represent the crowd work outsourced by the requester in our system as a workflow-based schema. Specifically, we define it as a three-tuple: *CW(G, T, D)*, where:

- *G* is the goal of current crowd work;
- *T* is a set of tasks;
- *D* is the interdependencies among the above tasks.

The goal G denotes the intended outcome of the outsourced crowd work, including a description and a collection of expected output data. G can also be used by requesters to search similar crowd work that has been done in the past, which can help them define the current one.

The set of tasks T and its interdependencies D compose the main part of a schema, which provide a skeleton for us to bind appropriate providers and orchestrate them. For each task in T, there are two parts of information, which are workload and constraint, where:

- *Workload*: the input and output data of a task;
- *Constraint*: the constraint of performing a task, i.e. deadline, cost, and quality criteria.

These information can be used to bind a specific task with a particular provider, as the workload corresponds to the service definition in a provider's profile. Also, it can be used to monitor task performance and control the quality, since the constraint can be utilised for negotiation between the requester and provider to gain their commitment on task performance. In addition, the interdependencies D, e.g. control-flow, data-flow, etc., can be used to orchestrate the bound providers to complete the crowd work.

Both this workflow-based schema and the service-oriented profile in Sec. IV are the data model in our work. Next, we introduce a coordination protocol as our operation model in Sec. VI to explain how we use our data model to meet the requirements in Sec. I.

VI. Coordination Protocol of Crowdsourcing

In this section, we elaborate on a coordination protocol of crowdsourcing in our work to meet the coordination requirement in Sec. I. When a requester defines his/her work in the schema mentioned above and outsources it to the crowd, first we would find the qualified, trustworthy providers for the requester based on the given schema and providers'profiles. Then, we would get the commitment from both sides on the crowd work performance through their negotiation, and bind each determined provider to a specific task. After that, we orchestrate these bound providers and request their services to complete the outsourced crowd work. To be more specific, this protocol consists of the following phases.

A. Provider Finding

To find the right provider for the right task in the given schema, we compare the workload and constraint information of a task with providers'information via various services in their profiles. Therefore, we could see if any provider can match the outsourced task in terms of their qualification, motivation, and reputation.

Specifically, first we could see if a provider is capable of doing a task by matching the input and output of this task with any capability service definition in one's profile. Through

one's condition services, we could speculate that a provider may be interested in a task if its cost can match his/her required pay, and its deadline is within his/her availability. After that, we may find multiple candidates; at this point we can rank them by considering both service quality metrics and social network metrics, e.g. the average service rate and recommendation from a provider's track record services, the common social connections that a provider shares with this requester via information services, etc. In doing so, we could find and recommend those candidates who are qualified and trustworthy to a certain extent.

B. Provider Binding

After finding providers, we need to bind each selected one to a specific task before requesting them to produce the data. In other words, we need their commitment on the task performance before its execution, which is represented as an agreement signed by both the requester and bound provider through their negotiation. This agreement determines the final constraint information of a task and contains some detailed terms on task execution (i.e. *if..then* statements), e.g. if a task is finished, then its quality must be measured by a third-party service. Therefore, after binding, this agreement can be used to monitor or supervise the execution, e.g. sending notification when the deadline is approaching, calling the third-party service for evaluation when a task is done, etc. Further, a provider can be re-bound during the execution through re-negotiation with the requester on some agreement terms, e.g. extending the deadline. At this point, we would check if there is an agreement clash betweem tasks according to their dependencies in the schema, e.g. for two sequential tasks, the extended deadline of the first task becomes later than the second.

C. Task Execution and Provider Orchestration

When all required providers have been found and bound to specific tasks, it comes to orchestrate them to execute their tasks for the crowdsourced work. Fig. 3 shows the lifecycle of each task, during which its execution consists of:

- *service invocation*: this stage is to call the bound provider service to produce data for the task's output. After binding, each provider is automatically requested and their response comes asynchronously (as a human service) to create or update the output. A provider can provide multiple responses to one request for the output update, and the request can be a reminder without providers'response.
- *evaluation*: this stage is to evaluate the task's output when a task is finished by the bound provider. According to the binding agreement, evaluation can be done manually (i.e. by the requester), or automatically (i.e. by our system or the third-party service). Based on the evaluation result and detailed agreement terms, the task would be closed or redone.

As the crowd work is defined as a set of interdependent tasks, the execution of one task may drive another, based on their dependencies. Currently, we consider the interdependencies among tasks as control-flow that can be expressed by an executable workflow language. As we represent providers as services, we coordinate the execution of the whole crowd work in the way of orchestrating providers via a workflow engine.

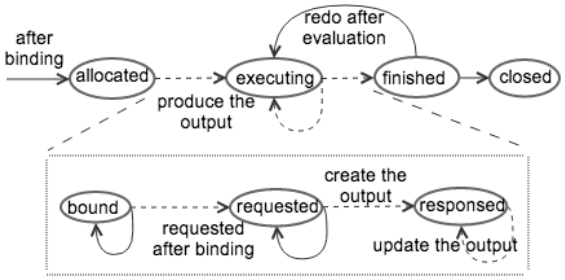

Fig. 3. Task lifecycle and its execution via service invocation: the above is task lifecycle and the bottom is service invocation for task execution.

VII. Conclusion and Future Work

We have presented an early conceptual framework for the crowdsourcing of complex work, which aims to bring the coordination support into current crowdsourcing platforms. Our framework is from the SOC viewpoint to treat crowd providers as services, which enable them to be discovered and composed into the crowd work on demand; meanwhile, we define a workflow-based schema to structure the complex crowd work, which provides us with a skeleton to organise and coordinate the crowdsourcing process. In doing so, we can find, bind, and orchestrate multiple providers to complete the complex crowd work. Our immediate future work centres around the implementation of the service-oriented profile, including the detailed representation in either WSDL or WADL, and integration with a workflow engine. Further, we need to detail the binding agreement representation and management.

References

[1] D. C. Brabham, "Crowdsourcing as a model for problem solving an introduction and cases," *Convergence: the international journal of research into new media technologies*, vol. 14, no. 1, pp. 75–90, 2008.

[2] M. Vukovic, "Crowdsourcing for enterprises," in *SERVICES 2009 - 5th 2009 World Congress on Services, September 21, 2009 - September 25, 2009*, ser. SERVICES 2009 - 5th 2009 World Congress on Services. Bangalore, India: IEEE Computer Society, 2009, pp. 686–692.

[3] A. Kittur, J. V. Nickerson, M. S. Bernstein, E. M. Gerber, A. Shaw, J. Zimmerman, M. Lease, and J. J. Horton, "The future of crowd work," in *2013 2nd ACM Conference on Computer Supported Cooperative Work, CSCW 2013, February 23, 2013 - February 27, 2013*, ser. Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW. San Antonio, TX, United states: Association for Computing Machinery, 2013, pp. 1301–1317.

[4] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: state of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, 2007.

[5] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut, "Crowdforge: Crowdsourcing complex work," in *24th Annual ACM Symposium on User Interface Software and Technology, UIST'11, October 16, 2011 - October 19, 2011*, ser. UIST'11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. Santa Barbara, CA, United states: Association for Computing Machinery, 2011, pp. 43–52.

[6] A. Kulkarni, M. Can, and B. Hartmann, "Collaboratively crowdsourcing workflows with turkomatic," in *ACM 2012 Conference on Computer Supported Cooperative Work, CSCW'12, February 11, 2012 - February 15, 2012*, ser. Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW. Seattle, WA, United states: Association for Computing Machinery, 2012, pp. 1003–1012.