Collaborative Infrastructure for On-Demand Crowdsourced Tasks

Julian Jarrett, M. Brian Blake
Department of Computer Science
University of Miami
Coral Gables, USA
{jarrett, m.brian.blake}@miami.edu

Abstract—Increasing popularity in the use of crowdsourcing has led to many tasks that can be fulfilled by the wisdom of human actors. When natural disasters or criminal activities occur, then sometimes crowdsourced tasks must be generated in real-time and must be fulfilled in an on-demand fashion. Effective use of crowdsourcing techniques requires an array of services that fulfill many dimensions of the overall problem such as resource selection and allocation, solution selection, and compensation. Any architecture that can provide these services in real-time, on demand requires a dynamic configurable infrastructure. This paper describes an adaptive framework for on-demand crowdsourced tasks supported by a design pattern-inspired architecture.

Keywords—crowdsourcing; elastic systems; service-oriented architecture; design patterns; collaborative infrastructure; crowdsourced tasks; online labor markets

I. INTRODUCTION

With the advent of the Internet, Web 2.0 and increased penetration of smart mobile devices, the popularity and utility of crowdsourcing has evolved. This paradigm creates new opportunities for the distribution of work when such tasks require human intuition and intelligence (human intelligence tasks, HITs) [1]. There is an increasing need for these HITs to be realized on demand; current infrastructure lacks dynamism and the configurable capability to address such a need. Currently, many standalone labor markets, such as oDesk and Amazon Mechanical Turk, exist on the Internet. They specialize in communicating, managing, and providing compensation for tasks that benefit from crowd interaction. These platforms, although effective, are limited to a subset of crowd workers who have self-selected their participation into the community. A more pervasive infrastructure might incorporate crowd interaction leveraging well-established, existing social and professional networking applications. Professional networking platforms such as LinkedIn, interests from social media profiles like Google+ or Facebook along with the previously mentioned labor markets themselves might provide a more comprehensive infrastructure. There are several open questions that must be addressed before such an infrastructure can be created.

- What are the general crowdsourcing services that any infrastructure must support?
- What design principles are necessary to support the underlying computational needs that underlie the general services?

In this paper we propose a descriptive architecture that realizes the services required for crowdsourced tasks (CT). This principled architecture leverages established design patterns and extensible techniques. Our primary contribution is the model for a general crowdsourcing infrastructure and a proposed framework that incorporates the internal services and distributed interfaces. First, we present a brief overview of crowdsourcing as a paradigm, followed by current platforms supporting crowdsourcing. Then, we described an operational crowdsourcing environment and derive the general services required for any supporting framework. Next we present our proposed architecture including our reusable design pattern-oriented architecture and its use for addressing elasticity and consensus tasks. We then provide a qualitative evaluation of our architecture before concluding.

II. CROWDSOURCING AS A PARADIGM

Crowdsourcing, a term coined by Jeff Howe [2], is the contraction of the words “crowd” and “outsourcing”. Typically, the paradigm involves large groups of anonymous individuals recruited to perform tasks [2]. The paradigm seeks to leverage the wisdom and expertise of individuals in the crowd through the subcontracting of work usually with an objective to lower costs and to scale and improve their products and services [3]. This notion of the working consumer is central to understanding crowd management services.

A. The Working Consumer

The role of the consumer has fundamentally changed from the traditional consumer to the emerging working consumer [3]. The consumers no longer merely purchase products and services but are now integrally involved in the value creation process [4]. They are given increasing roles and responsibilities within the production processes of firms as crowd input is valued as having tremendous commercial returns which far exceeds crowd compensation if any at all [3]. The paradigm also observes the reciprocal of roles between employees of a firm and the consumer; that is the working consumer is now the provider of a service or services of which employees are now clients [3].

B. Motivation for Crowdsourcing

Crowdsourcing has changed the notion of content creation. Traditionally, a small group of people created and controlled content for the consumption of the public. Crowdsourcing has now offered new opportunities for content to be created informally by the public for the public. Sites such as Youtube
and Wikipedia utilizing such approaches for content creation accounts for large percentage of traffic on the Internet with 10% of this traffic attributed to YouTube alone [5]. Motivation for this new phenomena is that a group of people provides a common good where there is a central regulatory absenteeism. Within the context of crowdsourcing this is perceived as access to content for public consumption that is typically not reduced through consumption [5]. Motivation to drive this informal content creation comes through several forms of compensation with most instances primarily using attention, trending topics and content hits sufficiently to forego monetary compensation [5].

In addition to value-added creation, firms also benefit from lower levels of risk. Firms have a reduced need for longer-term contracts with third-party service providers. Crowdsourcing minimizes the necessity to establish such dependencies through the fulfillment of services as a result of open calls via the Internet. Tasks are concurrently handled by multiple personnel in the crowd with varying levels of expertise. The company requesting the task however can accept or reject the outcomes of tasks only selecting those with results best suited to their needs and meeting desirable standards in quality. Only those working consumers who adequately fulfill the task receive compensation [6].

C. Crowd Management Services

There several services fundamental to the implementation and execution of the paradigm that need to be considered. These include concerns associated with crowd recruitment, compensation, optimizing costs and tradeoffs and finally managing and optimizing expertise/qualifying skills in the crowd.

1) Crowd Recruitment: The primary challenge with crowdsourcing is actually to attract and maintain a crowd [7, 8]. Recruitment is tightly coupled with compensation and infrastructure such as informal content creation sites and labor markets. Compensation can range from money to other varieties appealing to human centered psyche such as altruism, entertainment, attention, volunteerism [9, 10].

2) Compensation Models: Compensation models for crowdsourcing are typically seen in labor markets such as oDesk or Amazon Mechanical Turk [9]. This suits labour markets as some tasks are considered tedious and non-cash benefits are more difficult to quantify and adjust to the satisfaction of the working consumer [9, 10]. Depending on the distribution of work, different monetary compensation models exist. There are pay-for-performance models, quota systems with discretionary bonuses, team based compensation [10].

3) Costs and TradeOffs: Opportunities within crowdsourcing must address cost and quality concerns that require optimization. Often times, trade-offs must optimized cost, time and number of participants; authors in [11] argues that it is very difficult to impossible for the cost effective and timely acquisition of user input to have a positive impact on development. Other situations that have dynamic, scalable, on-demand, crowd sourced enabled labour force wrestle with issues of quality control [12].

4) Optimizing Skillsets and Expertise:

Requesters of services for work and tasks are challenged with identifying and optimizing skillsets and levels of expertise in the crowd. As such this requires a robust quality control infrastructure to address and mitigate errors [13]. A mechanism is needed to provide a tool to design practical tasks, refine and choose the pool of workers and disregarding completed tasks below the quality threshold or those incorrectly completed [12].

III. CURRENT PLATFORMS FOR CROWDSOURCING

Platforms facilitating crowdsourcing have been enabled by the evolution of Web 2.0 technologies. Some platforms handle specialized niche tasks while others accommodate more generalized tasks [14, 15]. Further advancement in smart mobile devices have provided new opportunities to engage the crowd-on-the-go. The mobile applications and embedded sensors provide accessibility and convenience [16].

A. Specialized Platforms

Specialized crowdsourcing platforms have leveraged Web 2.0 for tackling specific problems. These platforms maintain communities of specialist or persons interested in the problem space. These communities support task orientations ranging from software engineering, marketing and sales, knowledge communities, collaborative design, feedback research and innovation [17]. Authors in [18] outlined some actual implementations of specialized platforms currently in use. There are as follows:

- reCaptcha – decoding scanned text
- Galaxy Zoo – discovering and classifying celestial bodies
- Threadless – T-shirt designs
- Innocentive – Research and development

B. Labor Markets and Amazon Mechanical Turk

Amazon Mechanical Turk (AMT), a major Web 2.0 enabled platform, initiated services in 2005 [19] as a labor market tool [11]. AMT like other labor markets operate under the assumption that simple tasks requiring human intelligence (HITs) are better and easier performed by humans than their machine counterparts. Usually labor markets include some form of monetary compensation upon satisfactory and successful completion of HITs [3, 9, 11].

HITs are tasks posted by an owner or requester of work. The HIT is annotated with descriptions that include the nature of job, compensation, worker qualifications and skillsets, period of time that the worker may act on the task and the frequency that the HIT can be performed. HITs often provide an interface for the worker to perform the task; this is usually created by the requester [9]. A HIT usually carries a rate of compensation that often is proportional to the difficulty and the time requirements of the task. HITs are given a temporal, exclusive lock when workers offer to perform the task. Upon successful completion of the HIT within time constraints it is purged from the system; otherwise it still accessible for completion to other workers [19].

The low cost for participation in performing simple tasks is the driving factor for success for labor markets including AMT. They typically offer more cost effective alternatives than traditional methods. Traditional methods are better suited for more complex tasks, are generally more expensive and have lower numbers of participants. Another attractive characteristic
of labor markets is that they potentially offer a globally diverse worker base spanning time zones, geographic regions and cultures as opposed to traditional methods that can’t readily achieve this diverse scalability [11].

IV. A MODEL FOR GENERAL CROWDSOURCING SERVICES AND COMPUTATIONAL NEEDS

Considering the previously mentioned services, Figure 1 shows an operational environment that incorporates all of these services, in context. Any general model must have three major components, crowd interface, provider interface, and service coordination and middleware, and common communication channels. This model must also assimilate sophisticated computations methods listed in Table 1 and Table 2.

Crowd Interfaces. Open source, but sophisticated, crowd interfaces are currently limited. These specialized interfaces must integrate professional networks (e.g. LinkedIn), social networks (e.g. Facebook), and specialized crowdsourcing frameworks (e.g. Amazon Mechanical Turk). These interfaces will have different messaging needs. Professional networks may describe the crowd by their professional experience, but, in some sense, might also indicate expertise level considering years of employment. Social networks consider crowd interests and hobbies, but also users maybe credentialled by their associations. Specialized interfaces to standard crowdsourcing frameworks are relatively straightforward as the data exchange is already customized to working consumers.

Provider Interfaces. Industry, federal, and municipal organizations may have internal systems that establish statements of work or task orders. These task orders must be translated into jobs that can be realized by the crowd. The task orders will incorporate different types of information with varying formats.

Service Coordination and Synchronization and Communication Channels. The middleware layer provides service coordination and synchronization. As described in Section II.C, these crowd management services will consist of crowd worker recruitment services, management processes for compensating crowd work, persistent management when jobs are allocated, and sophisticated approaches of decision support to determine the best solution when the crowd produces multiple results. The middleware also contains system support for managing the various types of messages that must be transmitted. The messaging packets will consist of customer profiles, job profiles, job and solution data, and contracts for negotiated work.

The Service Coordination and Synchronization middleware layer (Table 1) will undoubtedly require computational techniques that will enable match-making. Some candidate approaches are case-base reasoning, collaborative filtering, and customized machine learning modules. Negotiating compensation quotes for crowd workers processes many attributes such as experience level and reputation and job complexity and criticality. The middleware will elastically allocated jobs and portions of jobs to different subsets of the crowd. Finally, information will arrive over time so the system will need to manage streams of information. The Specialized Interfaces (Table 2) will need to extensibly adapt to the evolving needs of the organizations and the ever-changing nature of the crowd.

<table>
<thead>
<tr>
<th>Crowd Interface (CI) Component</th>
<th>CI Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specialized Interfaces</td>
<td>Rapid Interface Extensibility</td>
</tr>
<tr>
<td></td>
<td>Flexible Communication, Publish/Subscribe, Unpredictable Alerts</td>
</tr>
<tr>
<td></td>
<td>Flexible Profile Data Management</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service Coordination and Synchronization (SCS)</th>
<th>SCS Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recruitment Management</td>
<td>Profile Matching, Collaborative Filtering, Case-Based Reasoning, Machine Learning</td>
</tr>
<tr>
<td>Compensation Processing</td>
<td>Multi-Dimensional Optimization</td>
</tr>
<tr>
<td>Job Allocation</td>
<td>Elasticity</td>
</tr>
<tr>
<td>Solution Resolution</td>
<td>Database Management, Stream Processing</td>
</tr>
</tbody>
</table>

V. REALIZING INTERFACE DEVELOPMENT THROUGH DESIGN PATTERNS

We propose a design pattern-inspired framework (Figure 2) to achieve elasticity, scalability and adaptability in our recruitment module. Under a façade, we implement the strategy pattern using an abstract singleton factory, a super-interface and the separated service interfaces for their corresponding external services to be plugins. All patterns operate and are coordinated under a web service driven 4-tier architecture and engaged by diverse clients through the Model-View-Controller pattern. In this section we briefly visit the patterns used and identify their roles within our proposed architecture.

The service layer (Figure 2 – 2.2), housing our strategy pattern built using the singleton abstract factory (Figure 2 – 2.2a), super interface (Figure 2 – 2.2b), separated service interfaces and plug-in, is our primary component offering dynamic behavior. The singleton aspect of our abstract factory forces the existence of one instance of this component. This optimizes on memory usage and allows the component to coordinate all services interfaces and their implementations. The framework’s runtime adaptability is attributed to dynamically changing task contexts. These contexts include metrics for scalability including budgets for tasks, desired levels of precision, time required amongst other domain-defined metrics [20] consumed by our strategy pattern hot swap services at runtime.

As shown in Figure 1, we consider four primary types of services; these are job allocation, recruitment, solution resolution and compensation services.
There services correspond to our specialized separated service interfaces and their respective concrete implementations / plug-ins as illustrated in Figure 2 showing the case of job allocation and recruitment services (Figure 2 – 2.2c and 2.2d). These interfaces allow for the connection from existing disparate remote external entities providing services for crowdsourced tasks or access to labor pools. We abstract all low-level workflow details using our façade layer (Figure 2 – 2.1) in our framework. This provides a consistent workflow protocol for all connecting clients to engage our recruitment service via high level function calls or wrapped API libraries. Finally the entire recruitment module is exposed to web, mobile and desktop client applications through endpoints via a uniformed web service interface (Figure 2 - 2) that is capable of delivering this dynamic behavior. The web service now acts as our model (Figure 2 – 1c) (data, data structures and data operations) for the corresponding view (Figure 2 – 1a) and controller (Figure 1 – 1b) (MVC) of the respective client software.

VI. ELASTICITY FOR CONSENSUS TASKS

Figure 3 illustrates an abstract overview of the elastic recruitment module’s interaction with external services. Community profile services may be professional networks (e.g., LinkedIn), resume repositories (e.g., Monster.com), social circle interests from social media (e.g., Facebook and Google+) and qualification profiles on existing labor markets (e.g. Desk and AMT). The job bank services primarily spans open labor market services such as AMT to closed groups of experts and professionals within a particular discipline. Our recruitment module is capable of scaling human computing elements from disparate sources to take up offerings for tasks in open labor markets or for those requiring specialized closed groups of experts. This architecture is not limited to simple crowd sourced tasks however can be extended to consensus tasks. Consensus tasks are those where the owner of the task is unaware of the solution however believes the solution exists somewhere in the universe and hopefully is answerable through harnessing the wisdom of crowds [21, 22, 23].

Figure 1 shows a more detailed interaction between existing entities tying the low-level implementation details in Figure 2 with the high level components expressed in Figure 3.
Figure 2. Design Pattern-Inspired Framework for Elastic Job Allocation and Recruitment.

Jobs are posted to the crowdsourcing job bank for the crowd to take up offerings. Again we see multiple entities from which we may engage the crowd those through professional and social networks and those through general and specialized labor markets. Professional networks deliver professional profiles and resumes to the crowdsourcing backend. Social network deliver interests in social circles and labor markets express an interest in work, worker availability and proposed rates for compensation.

All information form these sources are accessed through their respective API kits and pooled into a uniformed interface that allows the crowd sourcing backend to resolve several concerns; these concerns include job solicitation, resolution of suitable / candidate solutions and compensation to the workers. Once a worker in the crowd fits the profile to handle a particular job, the job profile, job request order and agreement are delivered to that worker. Given that a task may have several assigned workers or workers taking up the offering; solutions meeting the quality standards of the requesters must be met before it is accepted. Once the completed task satisfies these conditions, the worker is compensated based on the contractual agreement issued with the task.
VII. DISCUSSION

As illustrated in Figure 2, our framework utilizes a design pattern-inspired framework to coordinate multiple, disparate remote, pluggable services. Specifically, the strategy pattern coordinated by an abstract, singleton factory and our super interface abstraction, allows the framework to perform inversion of control (IoC) at runtime that allows for hot swapping capabilities, dynamically adjusting to potentially constantly changing contexts for on demand service delivery. Context can change with respect to the amount of workers required based on task complexity and worker constraint. This framework responds to dynamically changing runtime contexts. By abstracting the interfaces and pluggable services orthogonal design is achievable with higher levels of immunity to the underlying changes in external services. This design approach promotes reusability, maintainability and extensibility.

![Elasticity Architecture for Consensus Tasks](image)

Figure 3. Elasticity Architecture for Consensus Tasks.

VIII. CONCLUSION AND FUTURE WORK

As shown, we contribute a model that incorporates the many required services for crowdsourced tasks. Additionally, we propose a collaborative infrastructure to address on-demand crowd sourced tasks. We have presented a recruitment module utilizing an orthogonal design pattern-inspired framework to achieve elasticity (scalability), adaptability, maintainability and extensibility for the delivery of services. Our framework is capable of coordinating disparate, remote external software services from sources including repositories of work and an active labor force. For future work, we intend to implement a proof of concept and compare our framework’s performance against standalone labor markets such as AMT.

REFERENCES


