Crowdsourcing Crowdsourcing: Using Developers in the Crowd to Contribute Code to a Crowdsourcing Platform

Anand Kulkarni, Max Mautner, Brianna Salinas, Andy Schriner, Egor Vinogradov

LeadGenius
anand | max | brianna | andy | egor@leadgenius.com

Abstract

Crowd members drawn from international communities often possess a variety of unexpected skills, including programming and software engineering. At LeadGenius, we found that crowd workers with technical skills independently and regularly developed useful tools to enhance their own workflows and disseminated these to other crowd workers. We compare the utility of allowing the crowd to design and develop tools to directing suitable members of the crowd to develop tools under direction. Next, we make design recommendations for how crowd workers can be usefully harnessed in the development of a crowd engine.

Introduction

In this work, we explore how individuals working within a crowd platform can be usefully harnessed to make technical improvements to the platform. The initial motivation for this work came from our observation that crowd workers in the LeadGenius platform [6] independently and without direction produced and disseminated a variety of tools to make their own workflows more efficient. Following that discovery, we recruited a number of crowd workers and attempted to incorporate them into our standard software development process to build improvements to the platform under our own direction. We share initial observations of the crowd-developed tools, and some initial results from our experiments in working with members of this crowd in contributing to the crowd platform.

Related Work

Crowd work has been moving steadily into more complex and sophisticated domains. Prior efforts to crowdsource programming activities have attempted to decompose complex programming activities into simple microtasks that can be carried out by an expert crowd [1]. Separate systems have verified the ability of crowd workers to provide feedback on their own task instructions, as well as the results of other workers [2]. Previous work at LeadGenius found that crowd workers could be discovered who possessed programming skills using a directed search algorithm [3]. Commercial tools have successfully disseminated simple interface QA activities as microtasks on services like Amazon Mechanical Turk [4]. Last, the practice of outsourcing software engineering work to contract teams is reasonably well-understood, but has different dynamics and challenges than working with online crowds [5].

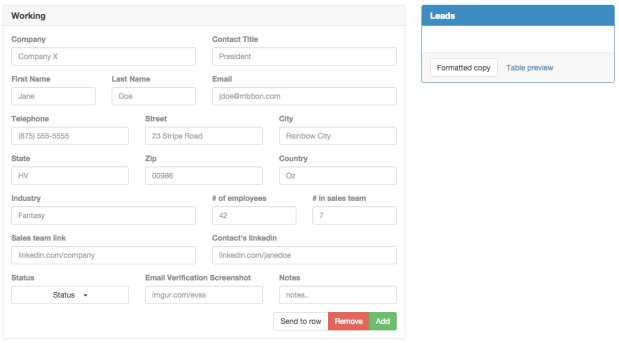

Figure 1: An example of a crowd-developed system designed, implemented and deployed by members of the crowd

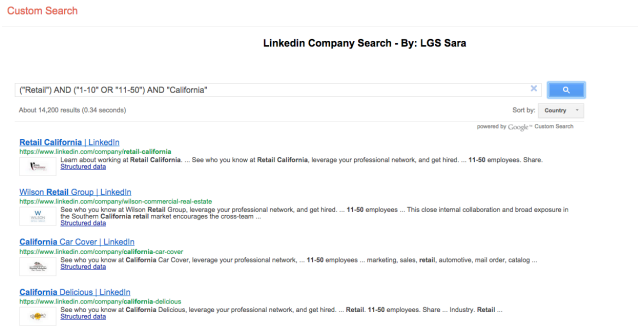

Figure 2: A custom search engine configured by a crowd worker.

Crowd-Produced Workflow Contributions

After discovering these tools in the wild as part of our regular community interactions, we examined the background of the crowd contributors to understand who was producing them. We interviewed two individuals. One individual was a Caribbean-based student of computer science who found crowd work opportunities doing simple data entry a lucrative side profession relative to local opportunities. He had multiple open-source contributions and was already familiar with Git, Javascript, and Angular. His primary interest was in making his own process of doing data entry more efficient, but that he did not do so with the explicit intention of improving the crowd platform overall. Another individual was a US-based individual who worked on Boolean term searches on library systems prior to joining the LeadGenius crowd. As a manager in the crowd community, her interest was explicitly in creating tools that could be used by multiple members of the crowd, sharing her knowledge, and in earning praise from other workers. In neither case was a direct incentive provided these tools.

Crowd-directed platform contributions

Javascript interface for deduplication (Figure 1): In this example, a crowd worker constructed and deployed (to a private server) a working Javascript application that accepted a set of data fields input by the end user and generated a formatted, deduplicated list. The formatting of the list enabled easy copy-and-pasting into the standard interfaces provided by LeadGenius. Qualitative discussions with workers indicated they found it more efficient than the standard interface.

Custom search engines for data research (Figure 2): In this example, a crowd worker used a combination of tools to produce both a set of customized Google search engines that extracted subsets of information from the internet, and a simple exporter that would export the results of these searches to a spreadsheet. Although the worker was not a sophisticated programmer, she read up on public resources, tested a variety of tools, and created a set of custom queries other workers could use.

Centrally-directed contributions

While it is useful for the crowd to discover its own solutions to problems they face, designers of crowd platforms may often find the tools produced by workers to work at cross-purposes with their own roadmap for a product. We attempted to have crowd workers contribute code to our core repository as a means of increasing our speed in building features already planned to be added to the LeadGenius system. We found that crowd workers fluent in programming were able to successfully run the LeadGenius system, learn our process for implementing changes and deploying code for review to in-house engineers, and make recommendations on changes that were being made. These workers were particularly useful in producing documentation and providing an additional perspective as a former user of the system; additionally, the output they produced had high credibility among other users in the system.

Design Recommendations

Set baseline incentives for emergent behavior: LeadGenius's core design, paying fair hourly wages, encouraging communication and incenting efficiency, incented workers to discover their own means of increasing productivity by making or combining tools without our explicit direction.

Use multiple workers to provide redundancy: Crowd workers tended to have highly intermittent availability and unexpectedly blocked production workflows when their internet failed. This can be avoided by the use of multiple workers on the same task.

Invest time in improving accessibility: Engineering systems in commercial use typically have complex production flows which need to be adapted to allow contribution from members of the crowd, accounting for security considerations and the variations in development environments and machines available to crowd workers. Resolving these considerations and bringing crowd workers represented a significant time investment before workers could contribute even minimally to a commercial codebase. It was critical to pair crowd members with a highly-available source of technical support inside an organization.

References

1. LaToza, Thomas D., W. Ben Towne, Christian M. Adriano, and André van der Hoek. "Microtask programming: Building software with a crowd." In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 43-54. ACM, 2014.

2. Dow, Steven, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. "Shepherding the crowd yields better work." In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 1013-1022. ACM, 2012.

3. Kulkarni, Anand, Prayag Narula, David Rolnitzky, and Nathan Kontny. "Wish: Amplifying Creative Ability with Expert Crowds." In *Second AAAI Conference on Human Computation and Crowdsourcing*. 2014.

4. www.rainforestqa.com

5. Bean, Michael. "The Pitfalls of Outsourcing Programmers." In *The Best Software Writing I*, pp. 9-15. Apress, 2005.

6. Kulkarni, A., Gutheim, P., Narula, P., Rolnitzky, D., Parikh, T., & Hartmann, B. (2012). Mobileworks: Designing for quality in a managed crowdsourcing architecture. *Internet Computing, IEEE*, *16*(5), 28-35.