ORIGINAL RESEARCH PAPER
Volume-6 | Issue-11 | November-2017 | ISSN No 2277 - 8179 | IF : 4.176 | IC Value : 78.46
INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

# CROWD SOURCING IN SOFTWARE –A STUDY

Computer Science

**Danda Swathi** M.Tech (Computer Science Engineering)

## ABSTRACT

The term crowd sourcing has entered software engineering practice. Internal development, contracting, and outsourcing still dominate, however crowd sourcing is a major source for software development projects for a various reasons, whether it is to squash bugs, test their software, or gather alternative designs for a new user interface. While the overall impact has been routine thus far, crowd sourcing has the potential to lead to fundamental and disruptive changes in how software will be developed in the future. This paper explores the models of crowd sourcing that have been applied to software development to date, outlines the exciting opportunities that exist, and articulates a series of challenges that must be overcome for crowd sourcing software development to truly reach its potential.

## KEYWORDS

Staffing, Software development and Programming management.

**Introduction:**
Crowd sourcing is the process of getting work or funding, usually online, from a crowd of people. The word is a combination of the words 'crowd' and 'outsourcing'. The idea is to take work and outsource it to a crowd of workers. Wikipedia is the best example. Instead of Wikipedia creating an encyclopedia on their own, hiring writers and editors, they gave a crowd the ability to create the information on their own. The result? The most comprehensive encyclopedia this world has ever seen.

**Crowd sourcing & Quality:**
The principle of crowd sourcing is that more heads are better than one. By canvassing a large crowd of people for ideas, skills, or participation, the quality of content and idea generation will be superior.

**Different Types of Crowd sourcing Models:**
*Model:* The act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call The above definition hones in on the three factors that distinguish crowd sourcing from other outsourced work: 1. the work being solicited through an open call to which basically anyone can respond, 2. the resulting engagement of workers who are unknown to the organization needing the work done, and 3. the potential for the group of workers to be large. These factors delineate crowd sourcing, though the exact nature of the open call and how it is issued, how an overall task is or is not broken down into smaller tasks, if and how workers collaborate, and other such factors remain unspecified.

**Peer production:** One of the oldest and most well-known models of software crowd sourcing is open source. Tens of thousands contribute to software projects such as Linux, Apache, Rails, and Firefox. Open source software development is an example of *peer production,* a model in which control is decentralized and contributions are made without monetary reward.

**Competitions:** A second crowd sourcing model, the *competition,* has recently gained significant attention in software development. The competition has similarities to traditional outsourcing, in which a client requests work and pays for its completion, but differs in treating workers as contestants rather than collaborators.

**Micro- tasking:** Another model of crowd sourcing is *micro tasking*, in which work is decomposed into a set of self-contained 'micro tasks', which each can be completed in minutes and which together compose into a solution to a more complex task. Micro tasking is best typified by Amazon's Mechanical Turk (AMT), a general platform in which clients post batches of micro tasks that workers complete one - at -a-time. To ensure quality, micro tasks are often completed by multiple workers, with voting and other mechanisms used to select the best solution.

To compare crowd sourcing models and provide a sense of the overall space in which different such models exist, we identify eight foundational and orthogonal dimensions along which crowd sourcing models vary, building on other models of the use of crowd sourcing more generally. The following table briefly explains these dimensions.

| Dimension | Brief explanation | Scale |
|---|---|---|
| Crowd size | Size of the crowd necessary to effectively tackle the problem | Small to large |
| Task length | Amount of time a worker spends completing an individual task | Minutes to weeks |
| Expertise demands | Level of domain familiarity required for a worker to make a contribution | Minimal to extensive |
| locus of control | Ownership over the creation of new (sub)tasks | Client to workers |
| Incentives | Motivational factors that cause workers to engage with the task | Intrinsic to extrinsic |
| Task interdependence | Degree to which tasks within the overall workflow build on each other | Low to high |
| Task context | Amount of system information a worker must know to contribute | None to extensive |
| Replication | The number of times the same task may be redundantly completed | None - many |

With these dimensions, then, it becomes possible to describe a wide range of models for crowd sourcing software engineering. The table below does so for a concrete example system of each of the three models discussed (peer production, competitions, and micro tasking). Other such systems can be similarly captured. Verification games, for instance, transform formal software verification problems into a game-like experience to which non- experts can nonetheless contribute.

**Opportunities**
It is interesting to consider the forces driving the current emergence of crowd sourcing models, platforms, and environments, particularly in terms of their use within software development organizations. Many of the crowd sourcing models is relatively novel, and their long-term benefits and drawbacks remain poorly understood.

*Reduced time to market* - Increased speed of development is a frequent reason to engage in crowd sourcing.

Generating alternative solutions - Organizing work into self-contained tasks makes it possible for multiple workers to each independently complete the same task.

*Employing specialists* - Decomposing large software development tasks into smaller tasks enables greater flexibility in the use of specialist freelancers.

*Democratization of participation* - A definitional characteristic of crowd sourcing is the democratization of participation.

*Learning through work* - Beyond status and glory, a key reason developers choose to contribute to open source is to learn a new technology.

## Conclusions

Crowd sourcing, in its various forms, has already changed software development. Open source aside, the number of new crowd sourcing platforms, the number of workers who sign up and actively contribute, and the number of organizations actively experimenting with crowd sourcing all are indicative of a phenomenon that in many ways has crept up more than taken the industry by storm. Regardless, serious challenges must be overcome if crowd sourcing is to have the same kind of impact in software development as it has had in other fields. The nature of software has much to do with it. Software is a complex artifact that is not easily broken down into clearly articulated, self-contained, and rapidly understood and completed tasks. Rather, its intricate and invisible nature poses a challenge to crowd sourcing that we foresee will take many years to address. Even so, it is rare that truly foundational shifts take place in our field, and crowd sourcing has that potential. It is worthwhile for the community to develop a deep understanding of how and when crowd sourced work can be applied within software projects.

## REFERENCES

1. K. Stol and B. Fitzgerald, “Two's company, three's a crowd: a case study of crowd sourcing software development.” International Conference on Software Engineering (ICSE), 2014, pp. 187-198.
2. N. Stephenson, Snow Crash. Bantam Books, 1992.
3. S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, Foldit players, “Predicting protein structures with a multiplayer online game.” Nature, vol. 466, 5 August 2010, pp. 756-760.
4. K. Mao, L. Capra, M. Harman and Y. Jia, “A survey of the use of crowd sourcing in software engineering.” Technical Report RN/15/01, Department of Computer Science, University College London, 2015.
5. W. Dietl, S. Dietzel, M. D. Ernst, N. Mote, B. Walker, S. Cooper, T. Pavlik, and Z. Popović, “Verification games: making verification fun.” Workshop on Formal Techniques for Java-like Programs, 2012, pp. 42-49.
6. T. W. Malone, R. Laubacher and C. Dellarocas, “The collective intelligence genome.” MIT Sloan Management Review, vol. 41 (3), 2010, pp. 21-31.
7. J. Surowiecki, the Wisdom of Crowds. Random House, 2005.