Leveraging the Wisdom of the Crowd in Software Testing

Mukesh Sharma
Rajini Padmanaban

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20140805

International Standard Book Number-13: 978-1-4822-5448-8 (Hardback)

Library of Congress Cataloging-in-Publication Data

Sharma, Mukesh (Software testing engineer)
Leveraging the wisdom of the crowd in software testing / Mukesh Sharma, Rajini Padmanaban.
pages cm
Includes bibliographical references and index.
ISBN 978-1-4822-5448-8 (hardback)
1. Computer software--Testing--Data processing. 2. AskA services. 3. Internet research. I. Padmanaban, Rajini. II. Title.

QA76.76.T48S53 2014
005.3028'7--dc23 2014027273

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

Contents

Foreword

Thousands of years of history, dating back to the Paleolithic Era, have taught humans the value of working together. Several strong warriors were required to work together to bring down a wooly mammoth for food and clothing. Specialists needed to learn how to grow food or pray for crops, while others depended on their talents. Human history is full of examples of people working together to overcome seemingly insurmountable challenges. The building of the Great Pyramid of Giza or the Temples of Ancient Greece, curating the Hanging Gardens of Babylon, constructing the Freedom Tower in New York City through the accuracy of Wikipedia and the richness of Twitter and Facebook—all of these amazing accomplishments in human history came to be only because of the coordinated work of many. In my own home many years ago, the phrase "many hands makes light work" was met with groans, as it was shorthand for "kids help out with chores." However, as we have entered a 21st century with ubiquitous connectivity and an "always on" mentality, the idea of everyone being available to contribute some of their discretionary time to a common cause is not too far-fetched. In fact, it's becoming the norm. We saw the rise of SMS displaced temporarily by Farmville updates, and then on to WhatsApp, SnapChat, Instagram and Pinterest. People working together to share information and improve each other's lives through a collective effort. While it may be premature to dismiss

Garret Hardin's Tragedy of the Commons – in which individuals act selfishly contrarian to the best interests of a group – it's safe to say that technology has leveled the playing field and enabled a more collaborative model of collective contribution where everyone can "win" in their own unique way.

There are thousands of examples throughout human history to illustrate the value of a diverse group of people focused on a challenge. If we were to pore through that hypothetical list, I would imagine that much time would elapse before software testing came in to view. Fortunately, Mukesh and Rajini have done the heavy lifting to create a framework for all of us to use to successfully apply these collaborative crowdsourcing techniques to improve the quality of our software. This book is unique in explaining how to combine the power of crowdsourcing with the craft of software testing to share practical experience, lessons learned, and provide guidance for others to hit the ground running. The rich set of examples provided here helps set the context for all practitioners to help improve understanding of a fast moving field. The depth of experience comes through with guidance on how to build a platform, create an ecosystem, and avoid the many potential problems and pitfalls that present a threat to successful crowdsource testing initiatives. The Did-You-Know conclusion offers pragmatic suggestions and offers a synopsis of advice to move forward on crowdsourced testing.

Quality is an amorphous, ephemeral condition, especially in software—and very hard to achieve in an increasingly online and mobile world. High quality requires meeting the needs and requirements of a broad and diverse set of people—and to do so requires a diverse set of analysts—a crowd—to help ensure everyone's needs are met. This book offers a formula for helping everyone be successful in leveraging the crowd to improve the quality of the software experience.

Ross Smith
Director of Test, Skype Division, Microsoft

Overview

Many hands make light work.

None of us is as smart as all of us.

The market is always right.

These statements very succinctly reflect the sentiment of this book on leveraging the wisdom of the crowd in software testing. Crowdsourcing practices across domains can be traced back at least three centuries. Although it is such an age-old practice, it started taking formal shape in its current name only in the last decade, thanks to the penetration of the Internet, social technologies, the agile style of development, mobile and cloud computing, and the application-intensive software development focus. The crowdsourcing market was estimated to be US$500 million in 2011 and was projected to have an approximately 75% growth over 2010. Over two thirds of this growth was attributed to Internet services, media, entertainment, and technology.[1]

Global outreach, quick time to market, and a feature-rich design are some of the major drivers in today's market in determining a product's success. Product companies are constantly on the lookout for innovative development and testing techniques to take charge of these driving forces. One such paradigm software testing technique gaining popularity is crowdsourced testing. The scale, flexibility,

cost-effectiveness, and fast turnaround it has to offer are all being spoken about at length, in several forums. While there are some resources online to refer to, on what crowd testing is all about and how to leverage it, there is no one comprehensive book as of today on crowdsourced testing that talks about practices, case studies, and the future of this technique. This book is intended to fill that void and serve as the go-to material for anyone wanting to leverage the wisdom of the crowd in software testing.

Specifically, while there are several online resources to understand the concept of crowdsourcing and examples of how it works in its various forms, there aren't formal resources to guide a tester, a test team, or a management team on what it takes to implement crowdsourcing in a software testing effort. We have been researching a lot on this area at QA InfoTech, presenting our experiences in conferences such as StarEast and StarWest and implementing crowdsourced testing for our clients at relevant places to supplement the core team's testing efforts. All of these together have been a major source of inspiration to write this book, which will serve as a practical guide for anyone wanting to adopt crowdsourcing for their software testing needs. The book is comprehensive enough to talk about the history of crowdsourcing and crowdsourced testing, implementation practices, and future trends. It provides the reader with a holistic and practical view of the topic and talks about building a career in this space. Since it also covers future trends, this material will be applicable for readers into the future as well. As practitioner's in the software testing discipline, we hope to bring out in this book our experiences, including some niche points such as defect management specific to crowdsourced testing and building a career in crowdsourced testing, which we have gained over the years through hands-on implementation. The book is divided into 10 chapters, and in this overview, you will read a brief outline of what is covered in each of them.

Chapter 1: Introducing the Crowd

Crowd forms the crux of this book. *Leverage the Wisdom of the Crowd in Software Testing* starts off defining who forms the crowd and why the crowd is of particular interest to us. It then talks about the history or timeline of when the crowd gained significance and how books

such as *Leveraging the Wisdom of Crowds* have been instrumental in helping the industry understand this concept. It also talks about the history of crowdsourcing in the software product development world and the core characteristics or traits of a crowd that make it very valuable in this domain. Setting this baseline early on in the book is important to help you gradually move into understanding the scope of crowdsourcing and specifically crowdsourced testing.

Chapter 2: An Overview into Crowdsourcing

This chapter takes you from understanding what forms a crowd to understanding what crowdsourcing is. It talks about the definition of crowdsourcing, the surge in popularity of this concept, the varied forms of crowdsourcing with intuitive examples, and introduces you to crowdsourced testing. As part of crowdsourced testing, it also gives examples to help you understand the practical implementation.

Chapter 3: Why Leverage the Crowd for Software Testing?

From this chapter on, the book starts focusing heavily on crowdsourced testing. In this chapter, specifically, we talk about where quality stands as of today and why crowdsourced testing is relevant to the world of software quality at this time, as opposed to a few years back or a few years into the future. In this context, we also discuss a case study from Facebook on how it works with the crowd on an ongoing basis to test for its updates and new features. The focus also is on talking about various test attributes (functional, performance, security, globalization, compatibility, usability, accessibility, etc.) and discusses how crowdsourced testing is relevant to each of them.

Chapter 4: Is Crowdsourced Testing a No-Brainer Solution to All Quality Problems?

Crowdsourced testing, while very powerful and effective, has its own limitations, like any other solution. This chapter talks about the challenges, limitations, and situations when crowdsourced testing will not work. These constraints could be from technology, logistics, or effectiveness angles. Understanding these is important, as they form our

problem statement for implementing an effective solution that will be discussed in subsequent chapters.

Chapter 5: How to Successfully Implement Crowdsourced Testing?

Having talked about the constraints in leveraging crowdsourced testing, this chapter is the core and essence of the book, as it talks about how to successfully implement a crowdtest effort. It talks about best practices in implementation, answering several questions such as what, when, and how to crowdsource, in a test effort. It covers best practices in mitigating the constraints and challenges discussed earlier, including solutions such as a crowdsourcing platform that can be built to test software products in the cloud. Several case studies that discuss how crowdsourced testing was adopted in both product and services companies are elaborated on in this chapter. We also look at the various engagement models in which crowdsourced testing can be implemented.

Chapter 6: Defect Management in Crowdsourced Testing

Defect management is a beast of its own in software quality engineering. It makes or breaks the quality of the product and the reputation of the test team. While defect management has its own challenges even in a centralized test team, one can only imagine what it would entail in a crowdsourced test effort, where the team is most often de-centralized and does not have insights into the team's executional practices. This chapter solely addresses effective defect management in crowdsourced testing, discussing how to keep the effort simple yet effective so it is a win-win situation for both the crowd and the test organization.

Chapter 7: Win Your Team's Support in Implementing Crowdsourced Testing

A test team, although convinced about crowdsourced testing, can face hurdles in implementing it at various levels—from its stakeholders, from other team members, from end users, etc. This chapter focuses on identifying such blocks and how to successfully win team's confidence

in implementing a crowdsourced test effort. This is a process that might take time and several iterations. The overhead may also be high, all the way from talking with the stakeholders to addressing their concerns to doing pilots to convince them on sample results that the effort can yield. However, without the support of the stakeholders, a crowdsourced test effort may not even take off or may fail miserably even in areas where it can potentially scale well. Given the importance of this activity, one full chapter has been dedicated to this topic.

Chapter 8: Let's Understand Crowdsourced Testing Holistically

By now, after reading through the first seven chapters, you will have a very detailed idea on what crowdsourced testing is all about and how to implement it successfully. This chapter serves as a wrap-up note helping the tester step back from all the details and look at the landscape holistically to differentiate myths from facts and emerge with a clear understanding of crowdsourced testing. You will be able to draw the distinction by now as to what is a myth and what is a fact. This chapter also touches upon career opportunities that an individual has as a crowd tester, which will help you make useful and critical choices in one's career progression in the field of software testing. Thus, this chapter will serve as a refresher/summary helping you look back on the information from each chapter and serving as a useful guide into career planning in the field of crowdsourced testing.

Chapter 9: Future of Crowdsourced Testing

We get close to wrapping up this guide to crowdsourced testing by talking about the future of this area. Herein we look at who can leverage this technique—service companies, product companies, the general public, etc., what to expect in terms of trends in this area moving forward (say, two to three years down the line), what technologies will continue to give this practice its required facelift, and the forecasted market size of crowdsourcing and crowdsourced testing. Through this chapter, you will get not just a detailed understanding on implementing a crowdsourced test effort, but more importantly, a view into both the business and the engineering aspects of crowdsourced testing.

Chapter 10: Building an Ecosystem around Crowdsourced Testing

While all along in this book, you will have gained comprehension of what crowdsourced testing is and how to implement it in a project, this chapter, the conclusion, is slightly offbeat. Herein, we look at the need to build a crowdsourced testing ecosystem, who the players of such an ecosystem would be, and who would need to champion such an effort of building an ecosystem. This ecosystem is in itself a futuristic trend, and given its importance, we have dedicated one chapter to it. Given the scale and reach of crowdsourced testing, its value can further be maximized by building an ecosystem that brings together the crowd testers, seekers, a platform, a knowledge repository, and the right tools. That is the core focus of this chapter, gearing you to become an active participant in the development of such an ecosystem and getting you excited about the future of this discipline.

1

Introducing the Crowd

> Before the beginning of great brilliance, there must be chaos. Before a brilliant person begins something great, they must look foolish in the crowd.[2]

Software is no new term in the current-day world. It has evolved to revolutionize every discipline, both at an enterprise level and at a common man level. Technologies, engineering practices, team engagement, and collaboration models have continued to change over time to adapt to the need of the day, to bring in optimized solutions in delivering software on time, within the defined budget, and of exceptional quality.

As engineering practices evolve, so have the software testing practices that form a significant part of the larger set of development efforts. Software testing has been exposed to a lot of challenges in the last decade—if we were to see these challenges as opportunities, the scale, scope, tool set, and team level visibility of the testing discipline have increased manifold. Testing has had to keep pace with the newer technologies that the product team adopts and understand how they might impact testing to devise a holistic test strategy from manual and automated testing fronts. Testing teams are engaged in the product life cycle much earlier than they used to be, testing for functionality and compatibility across supported platforms; areas such as accessibility and usability are getting a lot more attention; globalization is an important piece of the testing pie; security now extends beyond the basic web application testing level. All of these have opened a whole new window of opportunities for the testing team to align with the rest of the product team.

As we talk about the changes in the software development world, one noticeable new change that the industry is embracing in recent years is "leveraging the crowd," be it in product design, development, or testing. To establish our baseline and a common understanding, let us focus in this chapter on what we mean by the term *crowd*, the history

of both crowdsourcing in general and crowdsourcing in product development, and the traits of a crowd that make it relevant to be engaged at various stages in the software engineering process.

What Is a Crowd?

There are varied definitions of this term that are floated around these days, especially after the term *crowdsourcing* became popular. While we will look at the varied forms of crowd and align them with the forms of crowdsourcing in subsequent chapters, we will herein look at a very simple definition of *crowd* to introduce the group to you. A crowd is a gathering of people, of varying numbers, who act in an extempore and independent manner. They are not necessarily tied to any organization, do not have any special accreditations, and typically lack organization. However, the term *crowd* could also be used to refer to a group of people with a common trait or characteristic. The origin of this word dates back to the year 1275 and has been a commonly used word since then. A few examples of using this term in a statement include:

- A crowd has gathered in front of the shopping mall.
- The elated crowd cheered the singer at the show.
- There was a large crowd from Berkeley in the SFO downtown to watch the fireworks.

The term *crowd* can also be used to refer to a group of things, though this is not a very common case in regular usage. For example:

- A crowd of new shops have opened in the Great American Mall in Phoenix.
- The marketplace is crowded with advertisements of various brands.

The word *crowd* is thus used in both its noun and verb forms to refer to both animate and inanimate objects to describe a large number of a certain thing.

History of Crowdsourcing

As you read this book, you will be interested in knowing that the term *crowdsourcing* is now officially part of the *Oxford Dictionary*.[3]

The term *crowdsourcing* is relatively new; it started gaining attention in 2006 when Jeff Howe coined this term in his article "The Rise of Crowdsourcing." Looking back at the history of the activity involved in crowdsourcing, which is leveraging the crowd to source information or solve a problem, the concept has been in practice long before the term started gaining popularity or use, in the current-day technology world. For example, one source dates the use of crowdsourcing to 1714,[4] when the British government announced a prize for anyone who could reliably calculate longitude. This article also cites other examples, including a competition in France initiated by King Louis IV for making alkali from sea salt. In the 19th century, the *Oxford Dictionary* used a crowd (of about 800 users) to catalog its words. All of these examples clearly show how crowdsourcing has been in use in varied countries and forms to solve issues that merely could not be done in-house with a team of experts.

In 2004, crowdsourcing started getting a lot of attention globally when James Surowiecki wrote a book called *The Wisdom of Crowds*. This is a very interesting book and one of the highly suggested reading materials for anyone who is new to crowdsourcing. As one of the early books written on this topic, this is still an authority in understanding why and how the crowd's collective intelligence is important in solving problems. James presents a lot of interesting examples in his book to explain the concept of crowd wisdom, but his introductory case on the weight of the ox is more than sufficient to summarize his case in point. He talks about an exercise where Francis Galton, in 1907, leveraged the crowd to find an ox's weight. Galton had 800 people guessing the weight of an ox after it was slaughtered. Of the 787 valid entries that he received, the average weight of the ox turned out to be 1197 pounds, while the actual weight was indeed very close, at 1198 pounds. This is a starter case with which James talks about the collective wisdom of the crowd and how it surpasses even individual inputs from experts (in this case, estimates from individual cattle experts). This book from James followed by the article from Jeff Howe created a lot of buzz for crowdsourcing between 2004 and 2006. This base they created has been instrumental in experimenting with the use of crowdsourcing in several domains and for a multitude of uses in the last eight years.

In case you wonder whether the examples quoted by James Surowiecki are the only ones that are backed by testimony of the power and wisdom of the crowd, there are other examples too to mention here. Recently, Gideon Rosenblatt, who was reading *The Wisdom of Crowds*, was inspired to try a similar exercise to validate the cases that James was talking about. He kicked off an exercise similar to guessing the weight of the ox—this time it was guessing the number of cereal pieces in a glass vase full of them.[5] Gideon leveraged the social computing technologies to reach out to the crowd in conducting his experiment; in this case, he used Google+. After an initial short trial with about 500 guesses, he increased the limits and got a total of 2238 valid guesses. He conducted the experiment one more time with a clearer image to ensure he was able to collect results that were reliable. The second time around he got 436 valid guesses. On analyzing the results further, he concluded that the median number of cereal pieces guessed by the first set of participants turned out to be 402, while from the second experiment the result was 450. The exact number of cereal pieces in the vase was 467. While the numbers here are not as precise as what they turned out to be back in 1907 when the ox weight guessing event happened, this additionally adds to our underlying school of thought over the course of this book that the collective wisdom of the crowd can be very powerful in eliciting answers, solving problems if the factors within which the crowd works are conducive to help them bring out their best. We will talk about those factors in detail in our subsequent chapters, but these examples will help form your base in understanding the power of the crowd.

History of Crowdsourcing in Software Product Development

In the previous section, we saw examples of how crowdsourcing has long existed before the term gained popularity and visibility among the masses. They were all practical problems that needed to be solved to create value to the general public or studies that were taken up to prove the power of the crowd. But what is the history of crowdsourcing in the world of software product development?

Beta testing is a very popular instance of crowdsourced testing. If we look at the history of beta testing, it dates back to the 1950s when IBM first coined this term to perform hardware testing at

the second level—a level outside of the team that develops the product. While IBM dropped usage of this term in the 1960s, the term had already taken such deep roots that several organizations started using it. So, in one sense the use of crowdsourcing, specifically crowd testing in the hardware world, started with IBM, making its way into the software world too. To this day, it is a strongly suggested and used testing technique to get product feedback from a select group of end users who have the domain knowledge and prior experience using the product's earlier versions. Beta programs were conducted not just for testing a built software to find existing defects, but also to get end user suggestions to incorporate in subsequent releases. The year that James released his book *The Wisdom of Crowds* Gartner published a report that Microsoft (specifically Windows) is the biggest beta tester in history.

So, while we are able to map the origins of crowd testing to the beta programs, how did crowd development efforts start? Herein, it is worth briefly looking at the history of open-source software development. The freeware programs started in the software world in the early 1980s when the free software movement was launched. It would interest you to know that even before the free software phase, open-source software existed in its own shape and form dating back to the 1950s. During this period, any software that was created was based on the principles of openness, where the source code would be distributed along with the software, empowering organizations to make code changes as needed to create functional software that aligns with their hardware and operating system (OS) needs. If we look at this in the current-day information technology jargon, open-source software was encouraged back then to allow organizations to take on the professional services activity themselves. So, this can be looked at as the origin of crowdsourcing in the software development world.

In all direct reference to the open-source movement, it started taking shape with the introduction of Linux in 1991. Linux is a very popular example of crowdsourcing in software development to this day and how volunteer developers work on contributing to the code base on an ongoing basis. While the work done as part of the Linux projects have a direct mapping to crowdsourcing from our viewpoint, it wasn't until 1998, when the Open Source Initiative was initiated, that this movement gained formal recognition.

From an implementation standpoint, these events give us a sense that crowdsourcing in the product development world, be it in the design, development, or testing phases, has existed since the 1950s, although it started getting formal recognition only in the last decade. And while organizations have leveraged it in possible ways over the last decade, there is still quite some ambiguity, fear of the unknown, lack of overall buy-in, need for better proof that crowdsourcing can be leveraged for a varied set of scenarios, etc., that still limit the industry-wide official acceptance of this concept. This book will be a step in that direction, de-mystifying a lot of such open questions and providing clarity with live examples of how organizations have benefited from crowdsourcing, specifically crowdsourced testing, encouraging better industry-wide acceptance and application of this concept in the coming years.

What Are the Traits of the Crowd That Make It Relevant to Be Engaged in Various Stages of Product Development?

The term *crowd* is magical—it has a simple meaning, but the power of the crowd is extremely high. A typical user of a product is also part of a crowd when you look at the overall group of end users. To that effect, if we were to look at the traits of a crowd that make it relevant to engage in product development (be it in the design, development, content creation, or testing stages), the following core points emerge:

End user mind-set: The crowd can be gathered from a part of the organization or from a representative base of end users, external to the organization. Depending on the type of product/application that is being built, if it is from an external representative user base, it brings very rich criteria to the table, which is the end user mind-set. This is a trait that the organization cannot match even if it brings the most expensive testers on board. For example, an organization that is building a K–12 math software decides to go to a nearby school, to have a crowd of students and teachers use the product for a specific period of time and provide feedback. The richness and practicality in the feedback from, say, a second-grader may not be matched by even a test architect on the team, given that the second-grader is a realistic end user.

This feedback from representative end users is used not just at the testing phase, but should more importantly be included in the subsequent planning and design phases as well, to build a product that aligns with end user needs.

Subject matter expertise: The crowd can be a group of people with the required subject matter expertise in working on a product to address the team's constraints, including lack of very specific subject matter experts and lack of time and budget to handle everything in-house. While one would expect the internal team to build the required subject matter expertise in building a product, it is not always feasible to have a fully staffed team working on the project at all times. There may be situations where experts are not available, finding the right expert may be an issue, and it may not make financial sense to have the expert in-house for the entire duration of the project. The crowd is a sought-after solution in such cases since one can find the right crowd on an on-demand basis to address the organization's domain expertise needs. For example, let's say an organization in North America is building a global product and has created localized versions of its solution. It does not have the right expertise to verify the content in specific languages, for example, Balinese. The requirement here may not be long term, but verifying the content before product release is an important step to ensure overall product success. Given the trait of the crowd where the representative group of people has the required subject matter expertise in this case, the crowd is a very relevant solution to leverage. Similarly, the crowd can be a community of open-source developers. For example, Linux is a popular case of community-contributed code for the open-source operating system. The crowd herein has very rich subject matter expertise that is helping it contribute to the source code development as well as code verification.

The testing attitude: This trait of the crowd is specifically aligned to the product's testing needs. Every user (that together forms the crowd), when chosen as a representative base for a product, will bring an important element of the testing attitude to the table. For example, if you are an avid smartphone user,

you will have inherent traits of a tester in the smartphone market, such as the ones below:

- **A thirst to understand how things work:** You will be interested in knowing how things work, and to understand this, you will play around with the product proactively, think of ways to improve user experience, share your thoughts in relevant forums, be excited about opportunities where you can beta test the product, etc. This analysis mind-set often makes the crowd a great testing team.
- **A sense of inquisitiveness/curiosity:** In line with the point above, the crowd is a curious bunch of people that wants to understand what options are available in the market, what features are coming out in the next release, how this product fares against competition, etc. Such curiosity makes the crowd a very valuable group of people with a testing mind-set in evaluating the product.
- **An enhanced set of observations:** The crowd typically does not have full visibility into the system's internals. As an external group that validates and verifies the product or rather uses the product from realistic end user angles, the product team is able to elicit a more enhanced set of observations than what it could generate from within the team.

- **A questioning mind-set:** This is a trait in succession to how things work and the curiosity pieces discussed above. The crowd typically has a questioning mind-set, where it does not want to accept anything at face value. It wants to question claims made by an organization, which is a great trait for eliciting better and richer product feedback. For example, the organization may make claims about the performance of a product, its page load time, response time, etc. The crowd will not rely on these numbers. It will actually gauge what the product performance is, at run time, to determine whether it is acceptable or not.

Using crowdsourcing in software product development is only a part of the overall pie. Crowdsourcing stretches itself into various other

forms/facets in engaging with users in solving a range of problems. The examples we discussed in this chapter, especially around the history of crowdsourcing, give us an understanding of the early use of crowdsourcing to solve community problems and invent new solutions. In continuation to those, in the current day, we have specific forms of crowdsourcing that are leveraged not just to benefit software organizations, but more importantly to bring in the crowd to help create solutions that are important to the community—for example, in areas such as education, healthcare, societal uplift, etc. These are the forms and the varied manifestations of crowdsourcing that we will see in Chapter 2, which will also then set the base for us to start delving into the specifics of crowdsourced testing.

Did You Know?[D-1,D-2]

- Crowdsourcing is also referred to by other names, such as *fan sourcing*, *crowd casting*, *open sourcing*, and *mass collaboration*.
- Examples of who uses crowdsourcing include Amazon, Netflix, Wikipedia, and DuPont.
- Estimates say that about 1 million workers have been paid $1–2 billion for crowdsourcing projects.
- A crowdsourcing contest for Coca-Cola's energy drink brand, Burn, yielded 135 rich video entries in just 5 weeks. Here are its tips for successful crowd content creation:
  - Don't plan for a viral campaign.
  - Do it with consumers.
  - Aim for strong emotional reactions.
  - Seed! Don't wait for people to discover it.
  - Create a consistent and shareable experience.
- Get your timing right.