

Qing Wang
Zhenyu Chen
Junjie Wang
Yang Feng

Intelligent Crowdsourced Testing



Springer

Intelligent Crowdsourced Testing

Qing Wang • Zhenyu Chen • Junjie Wang •
Yang Feng

Intelligent Crowdsourced Testing

 Springer

Qing Wang
Institute of Software
Chinese Academy of Sciences
Beijing, China

Zhenyu Chen
Software Institute
Nanjing University
Nanjing, Jiangsu, China

Junjie Wang
Institute of Software
Chinese Academy of Sciences
Beijing, China

Yang Feng
Software Institute
Nanjing University
Nanjing, Jiangsu, China

ISBN 978-981-16-9642-8 ISBN 978-981-16-9643-5 (eBook)

<https://doi.org/10.1007/978-981-16-9643-5>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Foreword

At every minute of every day there are probably hundreds of millions of computers that are idle. It is staggering to contemplate the enormity of the amount of computing capacity sitting idle and unused the totality of all of these devices. And it is intriguing to consider what might be accomplished if even a modest fraction of that capacity could be put to good use. Indeed, others have not only contemplated such usage but some have also tapped into it. In one such project, that computing power is used to analyze electromagnetic spectrum for possible emissions from extraterrestrial civilizations. Another project attempts to use that computing power to evaluate approaches to predicting the structure of the proteins expressed by a gene, given only the DNA sequence of that gene.

In this fascinating book, the authors suggest how all of that unused computing power could be put to use by software engineers, using it to perform otherwise unachievably thorough testing of software through crowdsourced testing. The prospect of using the combined computational capacity of millions of idle computers all around the world to carry out unprecedentedly thorough testing, thereby improving the quality of the world's software, is exciting and intriguing. It is the subject of this book.

While the upside potential of crowdsourced testing is enormous, the difficulties in doing this are also enormous. They range from deciding how much capacity is available on which computers to how to apportion testing tasks to each available computer, to deciding how to integrate all of the testing results that have been returned, to knowing what to do when intentionally overlapping testing tasks have returned inconsistent results. This book addresses all of these problems, and more. In doing so, it makes for fascinating reading and contemplation, and also lays out a challenging and invigorating research agenda.

I congratulate the authors for this important and intrepid undertaking. Their vision is broad and exciting, and their research roadmap is challenging and stimulating. This book seems destined to become a seminal work in an area of boundless importance and promise.

Orleans, MA, USA
15 October 2021

Leon J. Osterweil

Preface

Software is everywhere today. It leads our every step. It is part of everything we do. Software makes our everyday work easier and simplifies our daily lives. We use software to work, study, and communicate with friends. It allows us to shop, make payments, travel, and do a lot more.

Yet when one programs, one makes mistakes. Every 1000 lines of code easily contain up to 16 errors, and a company's software has millions of lines, so that is a lot of errors. Once the software errors reveal themselves, they have consequences that range from "annoying" to "very severe." There has been a long list of software errors that have caused big disruptions. Airport systems can't function for a day, banking systems of entire countries shut down, and spacecrafts explode, among other mishaps.

There is no way to prove that a piece of software is 100% bug free. Nevertheless, there are things we can do to improve software quality. Among which, software testing is the most important strategy. Software testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free. If there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security, and high performance, which further results in time saving, cost effectiveness, and customer satisfaction.

Software testing involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. Traditionally, it was conducted by dedicated quality assurance teams with formally trained testers. Although these quality assurance teams are reliable, the high cost and delayed responses made them hard to scale and non-flexible for rapid update needs for the software industry today. Automated testing could be one solution, but the inability to create realistic user behavior test cases makes them hard to rely on given the variations in software products.

Crowdsourced testing is an emerging practice that enables testing with more flexibility and scalability than quality assurance teams. Crowdsourced testing, also known as crowdtesting, is a fresh approach to quality assurance. It combines human skills with technology to eliminate some of the problems involved in conventional

testing. Instead of carrying out testing within an organization, crowdsourcing uses a dispersed, temporary workforce of multiple individual testers. This on-demand community of testers is able to test the software more quickly and effectively than an in-house team. Crowdsourced testing offers companies an opportunity to have their products tested by real users on real devices across the globe, ensuring a customer-centric emphasis.

Thanks to the advantages of crowdsourced testing, it has been adopted by a growing number of organizations, including, but not limited to, Google, Facebook, Walmart, PayPal, and Uber. In particular, Google has deployed crowdsourced testing on its 14 major software product lines. Crowdsourced testing schema has also spawned a number of crowdsourced testing platforms. For example, Applause, which is the pioneer of global crowdsourced testing schema and the largest crowdsourced testing platform, provides usability, compatibility, security, functionality, accessibility, and other types of testing services. Synack, which is the world's largest secure crowdsourced testing platform, provides crowdsourced testing services to the U.S. Department of Defense, one-third US banks, and three-thirds credit card companies. The crowdsourced testing market is expected to register a compound annual growth rate of 10.7% over the forecast period 2021 to 2026.

Despite this increasing prevalence, crowdsourced testing is a new testing schema with unmaturing operational strategies and underdeveloped supporting technologies.

Meanwhile, intelligent techniques have seen successful in addressing various software engineering problems, for example, code generation, code recommendation, and bug fix and repair. Especially, artificial intelligence can be leveraged to enhance software quality assurance efficiently, for example, testing automation tools, guiding human testers in improving test coverage.

Benefiting from the rapid development of artificial intelligence, this book employs intelligent algorithms to facilitate various activities in crowdsourced testing. It provides supporting technologies in terms of the crowdsourced testing task, the crowd workers, and the testing results, which can increase the bug detection efficiency, reduce the manual effort, and potentially attract more crowd workers to promote the prosperity of a platform.

The aims of this book are to present the state-of-the-art technologies of using artificial intelligence algorithms to improve the crowdsourced testing activities, to provide actionable guidelines for industry practitioners, to inspire the researchers, and to encourage further research in this important and challenging area. This book is written for both software testing related researchers and industry practitioners. Researchers who want to obtain the knowledge of crowdsourced testing can find how we utilize artificial intelligence algorithms to improve various crowdsourced testing activities, and get inspiration in their own field. Industry practitioners of crowdsourced testing can use the introduced technologies of this book to upgrade their testing practice and improve testing efficiency. Besides, industry practitioners of other forms of software testing can also borrow the ideas presented in this book to facilitate similar tasks in their own scenarios.

We thank the researchers and students in the Laboratory for Internet Software Technologies, Institute of Software Chinese Academy of Science, and those in

the Laboratory of Intelligent Software Engineering, Nanjing University, for their dedication to the area of crowdsourced testing. We also thank the staff at MoocTest, China Software Testing Contest, and IEEE International Contest on Software Testing for their contribution on the experimental dataset.

We wish you interesting and enjoyable reading. We hope that you will benefit from this book and that you will be able to make use of the values of crowdsourced testing for accelerating and promoting the quality assurance activities.

Beijing, China
Nanjing, China
Beijing, China
Nanjing, China
October 2021

Qing Wang
Zhenyu Chen
Junjie Wang
Yang Feng

About the Authors

Qing Wang is a researcher at the Institute of Software Chinese Academy of Sciences (ISCAS). She is also the deputy chief engineer of ISCAS and director of the Laboratory for Internet Software Technologies at ISCAS. She currently serves as a director of the Board of Directors of the International Software and Systems Processes Association (ISSPA), a member of the International Software Engineering Research Network (ISERN), a member of the editorial boards of *Information and Software Technology Journal* (IST) and the *Journal of Software Evolution and Process* (JSEP), and a CMMI Lead Appraiser. She has served as the general chair of ESEM in 2015, and the program chair of ICSP from 2007 to 2009. Her research lies in the area of software process, software quality assurance, requirement engineering, knowledge engineering, big data, and artificial intelligence for software engineering. Qing has 20 years of experience in software process and quality assurance technologies. Her recent research related to software process and quality management has won the second prize of National Progress in Science and Technology of China and second prize of Progress in Science and Technology of Beijing. She has edited/co-edited 5 books and published more than 100 papers in international high-level conferences and journals.

Zhenyu Chen is the founder of MoocTest (moocTest.net), and he is currently a professor at the Software Institute, Nanjing University. He received his bachelor's and doctoral degrees in Mathematics from Nanjing University. Zhenyu worked as a postdoctoral researcher in the School of Computer Science and Engineering at Southeast University, China. His research interests focus on software analysis and testing. He has more than 100 publications in journals and proceedings, including TOSEM, TSE, JSS, SQJ, IJSEKE, ISSTA, ICST, and QSIC. He has served as the associate editor of *IEEE Transactions on Reliability*; PC co-chair of QRS 2016, QSIC 2013, AST2013, and IWPD2012; and program committee member of many international conferences. He also founded NJSD (Nanjing Global Software Development Conference). Zhenyu has won research funding from several competitive sources such as NSFC. He owns more than 40 patents (22 granted), and

some of his patents have been transferred into well-known software companies such as Baidu, Alibaba, and Huawei.

Junjie Wang is an associate researcher at the Institute of Software, Chinese Academy of Sciences (ISCAS). She received her PhD degree from ISCAS in 2015. Junjie was a visiting scholar at North Carolina State University from Sep. 2017 to Sep. 2018 and worked with Prof. Tim Menzies. Her research interests include crowdsourced testing, mining software repositories, and intelligent software engineering. She has more than 20 high-quality publications and has received the ACM SIGSOFT Distinguished Paper Award at ICSE in 2019 and 2020, respectively, as well as IEEE Best Paper Award at QRS in 2019.

Yang Feng received his bachelor's and master's degrees in software engineering from Nanjing University in 2011 and 2013, respectively. He obtained his doctoral degree from the University of California, Irvine. He has published more than 30 refereed papers and regularly serves as PC member and reviewer for international conferences and journals. His current research interests lie in software testing, crowdsourced software engineering, and program analysis.

Contents

Part I Preliminary of Crowdsourced Testing

1	Introduction	3
1.1	Why We Need Crowdsourced Testing.....	3
1.2	Benefits of Crowdsourced Testing.....	4
1.3	Current Practice of Crowdsourced Testing.....	5
1.4	Challenges of Crowdsourced Testing and Solutions.....	6
2	Preliminaries	9
2.1	Crowdsourced Testing	9
2.1.1	General Procedure of Crowdsourced Testing	9
2.1.2	Important Concepts of Crowdsourced Testing	10
2.2	Basic Introduction to Artificial Intelligence Technology	11
2.2.1	Supervised Learning.....	11
2.2.2	Unsupervised Learning	15
2.2.3	Semi-Supervised Learning	17
2.2.4	Conclusion	22
2.3	Typical Applications of Artificial Intelligence Technology	22
2.3.1	Natural Language Processing	22
2.3.2	Image Understanding	23
3	Book Structure	25
3.1	How the Book is Structured	25

Part II Supporting Technology for Crowdsourced Testing Workers

4	Characterization of Crowd Worker	31
4.1	Exploration of Crowd Worker's Characteristics	31
4.2	Crowd Worker's Characterization	33
4.2.1	Data Preprocessing	33
4.2.2	Activeness.....	34
4.2.3	Preference	34

- 4.2.4 Expertise 35
- 4.2.5 Device 36
- 5 Task Recommendation for Crowd Worker 37**
 - 5.1 Introduction 37
 - 5.2 Learning-Based Personalized Task Recommendation 38
 - 5.2.1 Motivation 38
 - 5.2.2 Approach 39
 - 5.2.3 Experiment 45
 - 5.2.4 Discussion 50

Part III Supporting Technology for Crowdsourced Testing Tasks

- 6 Crowd Worker Recommendation for Testing Task 55**
 - 6.1 Introduction 55
 - 6.2 Multi-Objective Crowd Worker Recommendation 56
 - 6.2.1 Motivation 56
 - 6.2.2 Approach 58
 - 6.2.3 Experiment 64
 - 6.2.4 Discussion 72
 - 6.3 Context-Aware In-Process Crowd Worker Recommendation 74
 - 6.3.1 Motivation 74
 - 6.3.2 Approach 76
 - 6.3.3 Experiment 81
 - 6.3.4 Discussion 87
- 7 Crowdsourced Testing Task Management 91**
 - 7.1 Introduction 91
 - 7.2 Completion-Aware Crowdsourced Testing Management 92
 - 7.2.1 Motivation 92
 - 7.2.2 Approach 94
 - 7.2.3 Experiment 101
 - 7.2.4 Discussion 108
 - 7.3 Improving Completion-Aware Crowdsourced Testing Management with Duplicate Tagger and Sanity Checker 109
 - 7.3.1 Motivation 109
 - 7.3.2 Approach 111
 - 7.3.3 Experiment 114
 - 7.3.4 Discussion 121

Part IV Supporting Technology for Crowdsourced Testing Results

- 8 Classification of Crowdsourced Testing Reports 125**
 - 8.1 Introduction 125
 - 8.2 Domain Adaptation for Testing Report Classification 126
 - 8.2.1 Motivation 126
 - 8.2.2 Approach 128

- 8.2.3 Experiment 131
- 8.2.4 Discussion 139
- 8.3 Local-Based Active Classification of Testing Report 141
 - 8.3.1 Approach 141
 - 8.3.2 Experiment 144
- 9 Duplicate Detection of Crowdsourced Testing Reports 153**
 - 9.1 Introduction 153
 - 9.2 Combining Textual Description and Screenshot Information for Duplicate Reports Detection 154
 - 9.2.1 Motivation 154
 - 9.2.2 Approach 157
 - 9.2.3 Experiment 162
 - 9.2.4 Experimental Dataset 162
- 10 Prioritization of Crowdsourced Testing Reports 169**
 - 10.1 Introduction 169
 - 10.2 Test Report Prioritization with Diversity and Risk Strategies 170
 - 10.2.1 Motivation 170
 - 10.2.2 Approach 172
 - 10.2.3 Experiment 178
 - 10.2.4 Discussion 183
 - 10.3 Multi-Objective Test Report Prioritization Using Image Understanding 185
 - 10.3.1 Motivation 185
 - 10.3.2 Approach 185
 - 10.3.3 Experiment 192
 - 10.3.4 Discussion 198
- 11 Summarization of Crowdsourced Testing Reports 199**
 - 11.1 Introduction 199
 - 11.2 Crowdsourced Test Report Aggregation and Summarization 200
 - 11.2.1 Motivation 200
 - 11.2.2 Approach 200
 - 11.2.3 Experiment 207
- 12 Quality Assessment of Crowdsourced Testing Cases 217**
 - 12.1 Introduction 217
 - 12.2 Assessing the Quality of Crowdsourced Test Cases Based on Fine-Grained Code Change History 218
 - 12.2.1 Motivation 218
 - 12.2.2 Approach 219
 - 12.2.3 Experiment 222
 - 12.2.4 Discussion 231

Part V Conclusions and Future Perspectives

13 Conclusions 235

14 Perspectives 237

 14.1 Fairness-Aware Recommendation 237

 14.2 Guidance in Crowdsourced Testing 238

 14.3 Data-Enabled Crowdsourced Testing 239

 14.4 Screenshots Learning 241

 14.5 Quality Assessment of Crowdsourced Testing 242

References 245

Part I
Preliminary of Crowdsourced Testing

Chapter 1

Introduction



1.1 Why We Need Crowdsourced Testing

The Internet is a decisive technology of the information Age, as the electrical engine was the vector of technological transformation of the Industrial Age. This global network of computer networks, largely based nowadays on platforms of wireless communication, provides ubiquitous capacity of multimodal, interactive communication in chosen time, transcending space. At the heart of these communication networks, the Internet ensures the production, distribution, and use of digitized information in all formats. The speed and scope of the transformation of our communication environment by Internet and wireless communication has triggered all kind of innovations around the world, among which crowdsourcing is the one that cannot be ignored.

Even though the practice of crowdsourcing can be track back to the late of 1990s, Jeff Howe firstly presents the systematic study on this topic in 2006 [30]. He described this new paradigm as “the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers”. By turning to a large group of people for ideas and solutions, crowdsourcing can generate a lot of benefits over internal ideation processes. Not only can businesses get access to great ideas, but they can also drive marketing buzz and engage their customers. People involved in crowdsourcing sometimes work as paid freelancers, while others perform small tasks voluntarily. For example, traffic apps like Waze encourage drivers to report accidents and other roadway incidents to provide real-time, updated information to app users. The benefits of crowdsourcing include: unexpected solutions to tough problems, greater diversity of thinking, reduced management burden, more marketing buzz, faster problem solving, and customer-centric data, etc.

Quality, in terms of validity and verifiability, determines the success of a software project. Many current software products are used by thousands or even millions of

people, helping them do their jobs effectively and efficiently or, alternately, causing them untold frustration and the costs of lost work or lost business. Testing is defined as “the process of executing a program with the intent of finding errors”. Hence, it is a destructive process of trying to find errors whose presence is assumed in a program. Its main goal is to establish a certain confidence that a program does what it is supposed to do. However, software testing cannot guarantee the complete absence of errors; instead, software testing attempts to be as complete as possible by identifying the largest possible number of errors. The best software is that which has been tested by thousands of users under thousands of different conditions. This is where the concept of crowdsourcing can help.

Crowdsourced testing, i.e., utilizing crowdsourcing paradigm for software testing, is the process of having a large group of individuals review one’s mobile apps, websites, mobile applications, software, products, or services to identify any defects or areas that can be improved before launching as part of the user acceptance testing stage. This form of testing, which is done remotely, differs from traditional in-house user testing by having the opportunity to branch out worldwide to receive feedback from multiple groups of people varying in age, sex, cultural backgrounds, etc. To receive feedback from testers, virtual machines or device emulators connected to a crowdsourced testing platform can be utilized, or, in some cases, testers may use their own devices to test any software, apps, etc. and submit their feedback. Once all the information is compiled and reviewed, developers can use the testers’ feedback to improve the applications or products they are working on.

1.2 Benefits of Crowdsourced Testing

Crowdsourced testing involves many individuals, often with better quality assurance and a reduced cost. Other major advantages that ensure an application is ready for launch include:

Speed—The importance of developers getting timely and relevant bug information is crucial. With such a large, global number of people involved in the testing, crowdsourced testing leads to fast execution and better results.

Efficiency—Crowd teams help some clients in time-varying bursts to manage peak workloads. This peak-demand testing strategy leads to the most efficient utilization of testing resources. Companies benefit from these burstable instances because they receive a high volume of quality assurance resources for a short period of time only when necessary, meaning they can manage day-to-day testing with fewer resources when necessary.

Cost-effective—To reduce labor costs and dramatically improve efficiencies, today’s software development teams tap into on-demand crowdsourced quality assurance services. It allows you to conduct testing without adding a permanent employee and the salary, pension, and other costly benefits incurred. Costs are dramatically reduced by applying the time-varying resource provisioning of managed crowdtesting.

Testing coverage—Crowdsourcing testing allows companies to attain the best test coverage as testing is performed on an extensive range of devices and platforms. This aids in unearthing bugs that might have been tedious to discover with conventional test mechanisms. The volume obtained via crowdsourcing testing assists in expediting the software product release cycle.

Geolocation—With testers located across the globe, it is much simpler for testing functions that depend on geolocation in crowdsourced testing. Testing in particular markets can also disclose any issues a website or application may have due to distinct cellular network or Internet speeds.

Usability testing capability—Through the power of crowdsourced testing, companies are able to determine quickly where their mobile application needs to be revised, updated, or fixed in order to avoid losing out on potential users. Crowdsourced testing delivers real users, on real devices, in their target markets.

While the pros of speed and flexibility of crowdtesting are remarkable, some issues can arise if one has not created a concrete test plan or rely on unsupervised testers from a marketplace. In addition, because payment usually depends on the number of bugs found and not their severity, testers may seek out and identify many small, less-important bugs rather than devote a lot of time and effort to finding one or two large, debilitating bugs

1.3 Current Practice of Crowdsourced Testing

Crowdsourced testing is an emerging paradigm that can improve the cost-effectiveness of software testing and accelerate its process. First, the project manager provides a test task for crowdsourced testing, including the software under test and test requirements. Then, the crowdsourced testing task is usually in the format of open call and incentives provision, so a large number of crowd workers can sign in to perform the task based on its test requirements, and are required to submit crowdsourced test reports.

Currently, crowdsourced testing has been adopted into the practice of a growing number of software organizations, including, but not limited to, Google, Facebook, Amazon, Microsoft, Alibaba, PayPal, Uber, McDonald's. Specifically, Google has deployed crowdsourced testing on its 14 major software product lines.

Meanwhile, crowdsourced testing schema has also spawned many startups that in turn propose plenty of methods, techniques, tools and platforms accelerating the development of crowdsourced testing. Applause (also known as uTest), which is one of the pioneers of global crowdsourced testing schema and the largest crowdsourced testing platform, provides usability, compatibility, security, functionality, accessibility and other types of testing services for Microsoft, Facebook, Disney, Wal-Mart, General Electric, Delta Air Services and thousands of other enterprises. Synack, which the world's largest secure crowdsourced testing platform, provides crowdsourced testing services to the U.S. Department of Defense, one-third of U.S.

banks, etc. Testing platform has more than a thousand different models of mobile phones, tablets, smart TVs and OTT terminals, and has provided crowdsourced testing services for more than 2.9 million applications. Almost all of the world top Internet companies, such as Facebook, Tencent, Alibaba, have organized their own crowdsourced testing team and built up crowdsourcing platforms. They not only employ crowdsourced testing for improving their products but also provide testing services and testing solutions for various end users.

According to Applause, the world's most popular crowdsourced testing platform, crowdsourced testing can increase testing capacity by 200%, increase the number of product releases per year by 150%, reduce the critical defect fixes needed by 50%, accelerate the planned revenue by 30%, increase the customer retention rate by 10% and conversion rate by 10%. In detail, through the rapid testing and feedback, quality assurance teams can better keep up with the increasing pace of development, which can potentially lead to the success of the product launch. The authentic feedback from real users in real-world settings helps a company improve app performance and deliver better customer experiences, which increases the customer satisfactory and their retention rates. And the remote, distributed teams help a company scale the test capability without the burden of additional overhead costs.

Due to the advantages of crowdsourced testing, the crowdsourced testing market is projected to register a Compound Annual Growth Rate of 10.7% over the prediction period 2021–2026. The main driving factors for the market comprise the increasing requirement of companies to enhance the user experience for competing in the current global market, increasing digital transformation, and building brand alertness, therefore aiding companies to adopt techniques to release their mobile apps or websites to the public fast.

1.4 Challenges of Crowdsourced Testing and Solutions

Despite of the aforementioned advantages, with the continuous development of crowd-sourced testing, the number of testing tasks, crowdsourced workers and test reports has increased dramatically, and the current crowdsourced testing still faces many challenges.

1. Testing task recommendation and management. Trade-offs such as “how much testing is enough” are critical yet challenging project decisions in software engineering. Insufficient testing can lead to unsatisfying software quality, while excessive testing can result in potential schedule delays and low cost-effectiveness. This is especially true for crowdsourced testing given the complexity of mobile applications and unpredictability of distributed crowdsourced testing processes. Experience-based decisions may result in ineffective crowdsourced testing processes, e.g., there is an average of 32% wasteful spending in current crowdsourced testing practices. Therefore, how to automate task closing

decisions and perform trade-off analysis is a critical research domain to improve the efficiency of crowdsourcing testing.

- 2. Characterization and task selection strategies of crowd worker.** Crowdsourced testing tasks are entrusted to the online crowd workers. Typically, a crowdsourced test task aims to detect as many bugs as possible within a limited budget. However not all crowd workers are equally skilled at finding bugs; Inappropriate workers may miss bugs, or report duplicate bugs, while hiring them requires nontrivial budget. Therefore, how to recommend a group of appropriate crowd workers for a testing task so that fewer workers can detect more software bugs has become an important challenge for the development of crowdsourced testing.
- 3. Test result analysis and management.** Crowd workers typically perform testing tasks and report their experiences through test reports. In the crowdsourced testing, the workers perform the tasks and then submit their test reports, which are simple and informal descriptions of the behavior of the software system. These test reports are composed of natural-language descriptions, sometimes accompanied with screenshots, and an assessment as to whether the worker believes that the software behaved correctly or behaved incorrectly. In a crowdsourced setting, the test reports are less structured and the number of test reports can be prohibitive. So, it is often impossible to manually inspect all test reports in a limited time.

In this book, we summarize our previous researches in the crowdsourced testing domain. For testing task recommendation and management, many researchers aim at exploring automated decision support to raise completion awareness crowdsourced testing processes, and manage crowdsourced testing practices more effectively. Particularly, in this book, we leverage dynamical bug arrival data associated with crowdsourced testing reports, and investigate whether it is possible to determine that, at certain point of time, a task has obtained satisfactory bug detection level. For characterization and task selection strategies of crowd worker, finding appropriate workers for particular software engineering tasks has long been recognized as being important and invaluable. With the emergence of crowdsourcing, there are several researches focusing on developer recommendation for crowdsourced software development. The aforementioned studies either recommend one worker or assume the recommended set of workers are independent with each other. However, in crowdsourced testing, a set of workers need to be recommended to accomplish a test task together. Furthermore, the recommended set of workers are dependent on each other because their performance can together influence the final test outcomes. In this book, we introduce some different approaches to recommend a set of crowd workers for crowdsourced testing tasks. For test result analysis and management, the aforementioned features of mobile crowdsourced test reports, i.e., high duplicate ratio, short text descriptions and rich screenshots, motivate us to propose some techniques to leveraging both the text and image information from duplicate reports to enhance developers' understanding of bugs, such as test report classification, duplicate detection, prioritization, summarization and quality assessment.

Chapter 2

Preliminaries



2.1 Crowdsourced Testing

2.1.1 General Procedure of Crowdsourced Testing

Figure 2.1 presents the overall procedure of crowdsourced testing. The project manager provides a test task for crowdsourced testing, including the software under test and test requirements. The crowdsourced testing task is usually in the format of *open call*, so a large number of crowd workers can sign in to perform the task based on its test requirements, and are required to submit crowdsourced test reports. The project manager then inspects these submitted test reports, confirm whether it is a bug, debug and fix it. Note that not every test report involves a bug, and different reports might describe the same bug (i.e., duplicate reports).

In order to attract workers, crowdsourced testing tasks are often financially compensated. The commonly-used payout schema includes paid by participation, paid by bug, and paid by first bug. Under *paid by participation* schema, workers are equally paid when they submit reports in a test task. It is mainly used for the newly-launched platform because it can encourage crowd worker's participation. Under *paid by bug* schema, only those crowd workers who detect bugs are paid (no matter whether it is a duplicate bug). Under *paid by first bug* schema, the crowd workers who detect the first bug are paid (the following duplicates would not be paid).

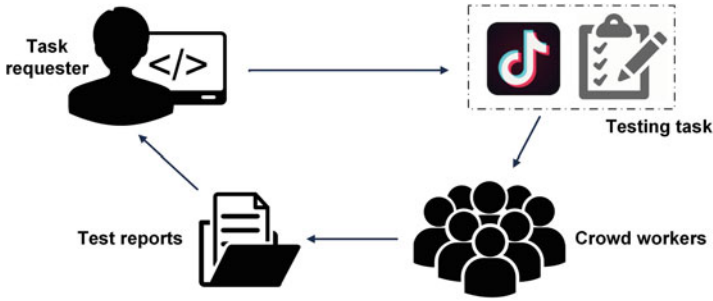


Fig. 2.1 The procedure of crowdsourced testing

2.1.2 Important Concepts of Crowdsourced Testing


We introduce three important concepts in crowdsourced testing: *Test task*, *Test report*, and *Crowd worker*.

A **test task** is the input to a crowdsourced testing platform provided by a task publisher. It contains task ID, task name, test requirements (mostly written in natural language), and the software under test (not considered in this work). Table 2.1 shows an example of a test task.

A **test report** is the test outcome submitted by a crowd worker after the test task is completed. It contains report ID, worker ID (i.e., who submit the report), task ID (i.e., which task is conducted), description of how the test was performed and what happened during the test, the screenshot of the bug. It can also include the bug label and duplicate label which are usually assigned by the project manager. Table 2.1 shows an example of a test report. Specifically, the labels are assigned by the project manager to indicate whether the report contains a “bug” (i.e., bug label), and whether the report is a “duplicate” of other reports (i.e., duplicate label). Note that, in the following sections, we refer to “bug report” (also short for “bug”) as the report contains bugs, while refer to “test report” (also short for “report”) as any report submitted in the test task (including bug reports and reports without bugs).

A **crowd worker** is a registered worker in the crowdsourced testing platform, and is described by worker ID, her/his context attributes (e.g., device model). The platform also records the worker’s historical test reports. A test task can be conducted by hundreds of crowd workers.

Table 2.1 An example of crowdsourced test task, test report, and crowd worker

<i>Test task</i>	
Task ID	T000012
Name	IQIYI testing
Requirement 1	Browse the videos through list mode IQIYI, rank the videos using different conditions, check whether the rank is reasonable
Requirement 2	Cache the video, check whether the caching list is right
<i>Test report</i>	
Report ID	R1002948308
Task ID	T000012
Worker ID	W5124983210
Description	I list the videos according to the popularity. It should be ranked according to the number of views. However, there were many confused rankings, for example, the video “Shibuya century legend” with 130 million views was ranked in front of the video “I went to school” with 230 million views
Screenshot	
Bug label	Bug
Duplicate label	R1002948315, R1002948324
<i>Crowd worker</i>	
Worker Id	W5124983210
Context	Phone type: <i>Samsung SN9009</i>
	Operating system: <i>Android 4.4.2</i>
	ROM information: <i>KOT49H.N9009</i>
	Network environment: <i>WIFI</i>
Historical	R1002948308, R1037948352
Reports	

2.2 Basic Introduction to Artificial Intelligence Technology

2.2.1 Supervised Learning

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

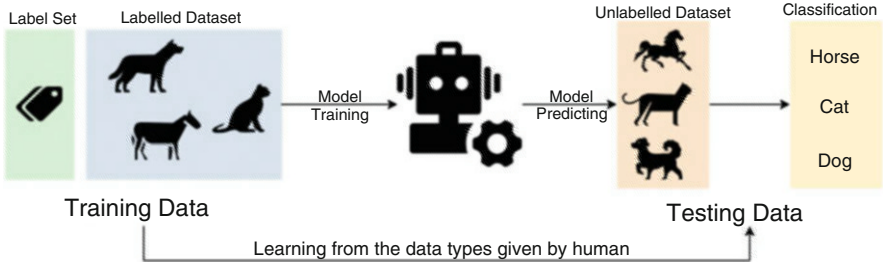


Fig. 2.2 An example of supervised learning

The working of supervised learning can be easily understood by the below example and diagram (Fig. 2.2):

Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a square.
- If the given shape has three sides, then it will be labelled as a triangle.
- If the given shape has six equal sides then it will be labelled as hexagon.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

2.2.1.1 Probabilistic Supervised Learning

Most supervised learning algorithms are based on estimating a probability distribution $p(y|\mathbf{x})$. We can do this simply by using maximum likelihood estimation to find the best parameter vector θ for a parametric family of distributions $p(y|\mathbf{x}; \theta)$. We have that linear regression corresponds to the family:

$$p(y|\mathbf{x}; \theta) = \mathcal{N}(y; \theta^\top \mathbf{x}, I). \quad (2.1)$$

We can generalize linear regression to the classification scenario by defining a different family of probability distributions. If we have two classes, class 0 and class 1, then we need only specify the probability of one of these classes. The probability of class 1 determines the probability of class 0, because these two values must add up to 1.

The normal distribution over real-valued numbers that we used for linear regression is parametrized in terms of a mean. Any value we supply for this mean is