

Software Testing

The State of the Practice

Mohamad Kassab, Joanna F. DeFranco, and Phillip A. Laplante,
Penn State Great Valley

// A web-based survey examined how software professionals used testing. The results offer opportunities for further interpretation and comparison to software testers, project managers, and researchers. The data includes characteristics of practitioners, organizations, projects, and practices. //



RECENT SOFTWARE-TESTING research has focused largely on designing new techniques and validating their effectiveness in real development contexts.¹ Throughout software development's history, however, testing techniques have struggled to keep up with the ever-changing trends in software development paradigms.¹ To trigger any improvement in this state of the practice will require serious effort in predicting the trends, learning stakeholder mind-sets, and pinpointing software-testing problem areas.

Software quality will likely become increasingly important in software marketing. As this situation

evolves, testing strategies will become progressively more important. Unfortunately, little data exists on software professionals' software-testing and quality assurance (QA) practices.

To discover these practices, we conducted a comprehensive survey of software professionals. We felt that such a survey could identify software-testing trends. The survey responses could also help others understand the relationship between areas such as software testing and quality.² In particular, we aimed to alleviate the lack of data and identify best practices, which could then be disseminated.

Survey Design and Implementation

We created a web-based survey using the QuestionPro survey tool (www.QuestionPro.com). We formulated the 40 survey questions after carefully reviewing similar surveys.¹⁻⁵ (For a brief look at other surveys related to testing, see the sidebar.) Designing the survey took approximately 120 person-hours over approximately one month. To allow a valid comparison with the other surveys, our survey included selected questions from those surveys. A summary of our questions is at goo.gl/kGBLhq.

The survey had six sections:

- the participants' professional information,
- the characteristics of the participants' organizations,
- project characteristics,
- integrating testing activities into the development lifecycle,
- methods and practices, and
- metrics and defect management.

We asked the participants to base their responses on only one software project that they either were currently involved with or had taken part in during the past five years.

The participants came mostly from a database of former software engineering graduate students from Penn State Great Valley, which caters primarily to working professionals. We sent an email invitation (and subsequent reminder) to them. We also posted an invitation to LinkedIn professional testing and quality groups to which one or more of us belonged.

We collected data from January to June 2015. Of the 199 people who viewed the survey, 167 started taking it. Of those survey takers, 67 (40 percent) completed the survey. The average time to complete it was

OTHER SURVEYS ON TESTING

Not much research has surveyed software professionals regarding testing. Jussi Kasurinen and his colleagues found that software professionals often considerably underestimated testing costs.¹ In addition, Kasurinen and his colleagues determined that more-effective testing might reduce testing time, which was also often underestimated.

Dudekula Rafi and his colleagues reported that automating testing could help improve test reusability, repeatability, and testing effort.² However, survey results have indicated a low percentage of projects that use automated testing.^{2–4} Other studies have indicated that tool adoption is also low.^{5,6} These results support our survey findings (see the main article). This might be due to the tools' high cost, the training required, and the test case design effort.² So, tool adoption can be a major barrier in testing.^{5,6} Clearly, organizations don't use testing tools effectively.⁷ These barriers might be why Kasurinen and his colleagues' survey indicated that testing constituted only an average of 25 percent of the total development effort.¹ On the bright side, even with these barriers, Vahid Garousi and Tan Varma found that the use of automated testing had increased between 2004 and 2009.⁶

Adnan Causevic and his colleagues' survey indicated that the participants used open source testing tools mostly for unit testing and used proprietary testing tools mostly for higher-level system testing.³ This corresponds with our survey results. Causevic and his colleagues also found that the survey participants didn't consider writing test cases before writing code to be a current practice. However, our survey showed that this approach is becoming established in practice.

A related research area is how to build effective testing teams. Tanjila Kanij and her colleagues surveyed practitioners to determine various factors' importance in building

testing teams.⁸ The results suggested that software-testing experience was more important than interpersonal skills. The results also suggested the desire for testing-team members to have diverse work experience.

References

1. J. Kasurinen, O. Taipale, and K. Smolander, "Software Test Automation in Practice: Empirical Observations," *Advances in Software Eng.*, vol. 2010, 2010, article 4.
2. D. Rafi, K. Moses, and K. Petersen, "Benefits and Limitations of Automated Software Testing: Systematic Literature Review and Practitioner Survey," *Proc. 7th Int'l Workshop Automated Software Testing Conf.*, 2012, pp. 36–42.
3. A. Causevic, D. Sundmark, and S. Punnekkat, "An Industrial Survey on Contemporary Aspects of Software Testing," *Proc. 3rd Int'l Conf. Software Testing*, 2010, pp. 393–401.
4. J. Lee, S. Kang, and D. Lee, "Survey on Software Testing Practices," *IET Software*, vol. 6, no. 3, 2012, pp. 275–282.
5. S.P. Ng et al., "A Preliminary Survey on Software Testing Practices in Australia," *Proc. 2004 Australian Software Eng. Conf.*, 2004, pp. 116–125.
6. V. Garousi and T. Varma, "A Replicated Survey of Software Testing Practices in the Canadian Province of Alberta: What Has Changed from 2004 to 2009?," *J. Systems and Software*, vol. 83, no. 11, 2010, pp. 2251–2262.
7. M. Grindal, J. Offutt, and J. Mellin, "On the Testing Maturity of Software Producing Organizations," *Proc. Testing: Academic & Industrial Conf.—Practice and Research Techniques (TAIC-PART 06)*, 2006, pp. 171–180.
8. T. Kanij, R. Merkel, and J. Grundy, "A Preliminary Study on Factors Affecting Software Testing Team Performance," *Proc. 2011 Int'l Symp. Empirical Software Eng. and Measurement*, 2011, pp. 359–362.

17 minutes. We also included the results of the incomplete surveys. We treated all responses anonymously and used only aggregate data, not the individual responses.

The Participants and Organizations

The participants represented 18 industries. They had an average of

7.8 years' professional experience. Sixty-eight percent of them had successfully completed a bachelor degree or equivalent, and 32 percent had a master's or doctorate.

Almost 44 percent of the participants worked in small companies (1 to 100 full-time employees); 30 percent worked at very large companies (more than 1,000 full-time

employees). The companies of 65 percent of the participants had an independent QA department—79 percent of the very large companies and 44 percent of the small companies. The companies were in nine countries.

The participants held diverse positions in the reported projects. Charts depicting the participant characteristics are at goo.gl/xWHEhO.

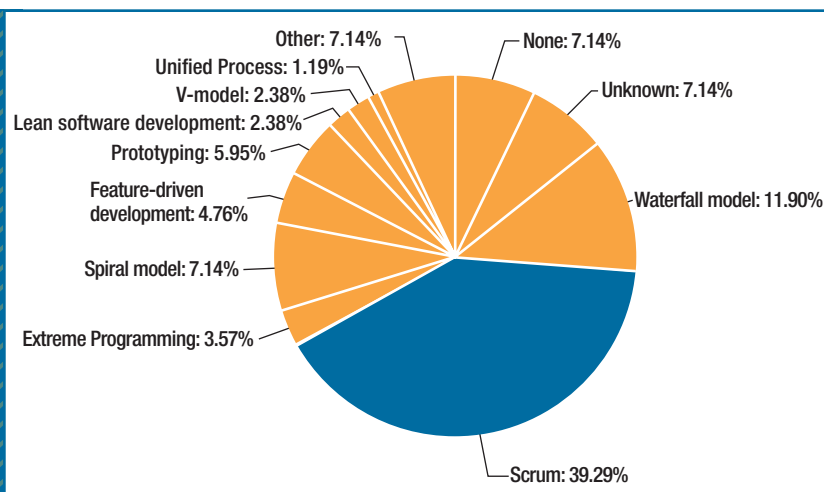


FIGURE 1. Participant responses regarding the development methodology that best described their project. Agile methodologies (for example, Scrum, Extreme Programming, or feature-driven development) were the most popular

Project Characteristics

The projects were distributed across a range of application domains, with a mild bias toward information technology (15 percent of the projects). The projects were also distributed across various categories, with a bias toward database projects (22 percent).

Sixty-five percent of the projects involved new development, 15 percent involved legacy system evolution, and 14 percent involved enhancement.

We also classified how safety-critical the projects were. The participants specified the maximum loss or damage if the software being developed failed (that is, the delivered service no longer complied with the specifications). Approximately 16 percent reported a highly critical system in which serious failure involved loss of life. Most participants reported the loss would be limited to essential funds, discretionary funds, or comfort.

Regarding the projects' duration from inception to delivery,

48 percent took up to one year, 21 percent took between 12 and 24 months, and 31 percent took more than 24 months. An average of 20 full-time staff (IT) were involved. In terms of LOC, the projects were predominantly of medium size (from 5,000 to 50,000 LOC); 36 percent contained 50,000 or fewer LOC.

Charts depicting the project characteristics are also at goo.gl/xWHEhO.

Integrating Testing into the Development Life Cycle

As software development has evolved into an engineering discipline, often involving distributed teams, the techniques and frameworks used have evolved accordingly. The need for cost-effective software production has also reached software testing. But how do organizations implement testing?

To answer this, we asked the participants which development

methodology best described their project. Agile methodologies (for example, Scrum, Extreme Programming, or feature-driven development) were the most popular, constituting approximately 50 percent of the projects (see Figure 1).

Forty-one percent of the participants who used agile development and 13 percent of those who used the waterfall model didn't have an independent QA team or department. However, this response must be seen in relation to company size; agile processes are popular mainly in small companies that usually don't have a separate QA team or department.

Most projects performed QA late during development, which is consistent with what Peter Haberl and his colleagues reported.¹ Only 19 percent of the participants performed QA in the study-and-concept phase, whereas 47 percent performed it during requirements specification and 49 percent performed it during system design. From implementation onward, QA practices increased significantly (see Figure 2). Projects using the waterfall model performed QA earlier than did projects using agile development.

Seventy-one percent of the participants agreed or strongly agreed that testing was a defined phase in their project. Eighty-four percent also preferred to have testing as a defined phase. Forty-four percent of the participants (47 percent of the agile projects and 30 percent of the waterfall projects) agreed or strongly agreed that testing and code development weren't separate phases in their projects, and 50 percent preferred not to have testing and code development as separate phases. If we consider dissatisfaction in a response as the difference between the current and preferred

practice, these numbers indicate a good level of satisfaction with how software testing is distinct from the other project activities.

Approximately 17 percent of the participants didn't explicitly estimate the size or effort for QA or testing activities (see Figure 3), compared to the 7.3 percent Haberl and his colleagues reported for a similar question.¹ When such estimation did happen, it occurred mostly together with the other development activities as a whole (37 percent of the participants). Only 29 percent reported that their team didn't do a good job of estimating testing activities' size or effort. Surprisingly, 43 percent said they didn't have enough time to test the software before deploying it.

Methods and Practices

The participants who had conducted testing activities answered questions about the techniques and tools they used.

Organizations often have defined levels of testing,³ including unit, integration, system, acceptance, and regression testing. Figure 4a shows that system-level testing was the most common (occurring in 81 percent of the projects). Approximately 90 percent of the projects performed system testing to test more than one system characteristic, with a clear focus on testing functionality (see Figure 4b). The survey presented a list of characteristics derived from Kshirasagar Naik and Priyadarshi Tripathy's system-testing taxonomy;⁶ participants could select all that applied. Performance was the most tested quality attribute. Seventy-eight percent of the participants primarily used black-box testing techniques; 42 percent used structure-based (white-box) techniques.

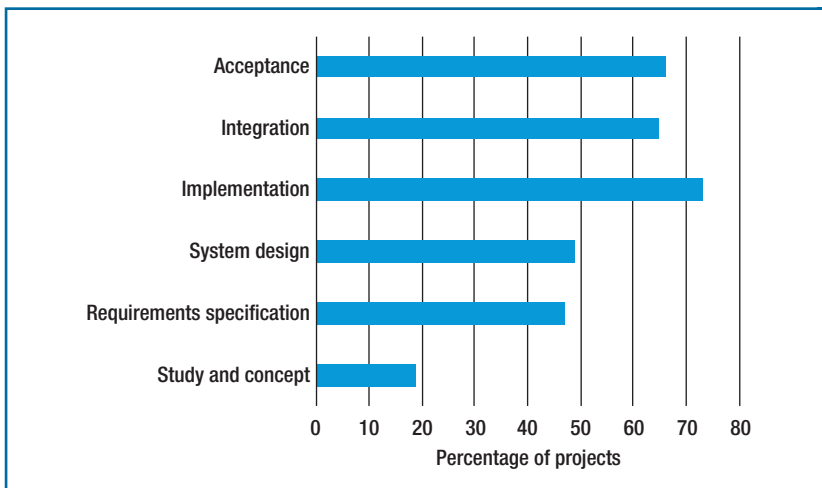


FIGURE 2. The development phases in which the projects performed quality assurance (QA). Most projects performed QA late during development.

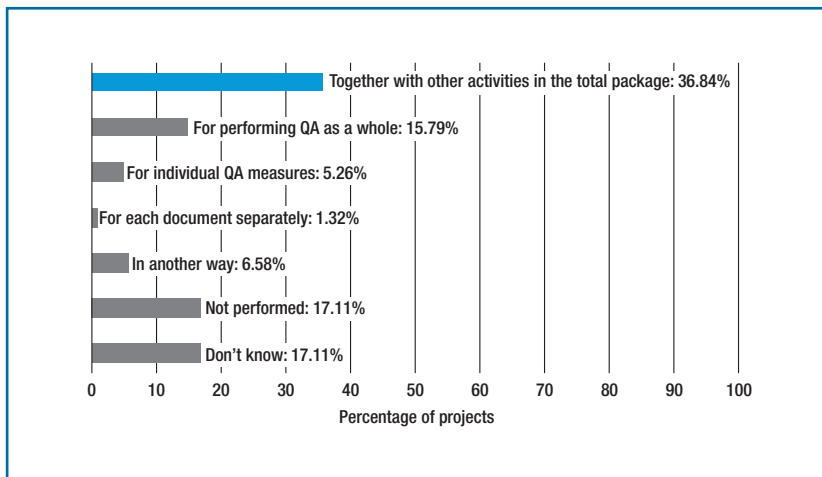


FIGURE 3. Participant responses as to whether and how the projects estimated the size or effort for QA or testing activities. Surprisingly, 43 percent reported they didn't have enough time to test the software before deploying it.

Regression testing seeks to uncover new bugs, or regressions, in a system's functional and nonfunctional areas after revisions such as enhancements, patches, or configuration changes. Only 12 percent of the participants reported that their projects outsourced regression testing; this result was close to the 11 percent who preferred to outsource regression

testing. This indicates a level of satisfaction with the current practice of regression-testing outsourcing. The most outsourcing occurred in the telecommunications and gaming industries (50 percent), compared to 33 percent for the IT industry.

The systematic design of test cases and definition of test data were widespread. Systematic test case design

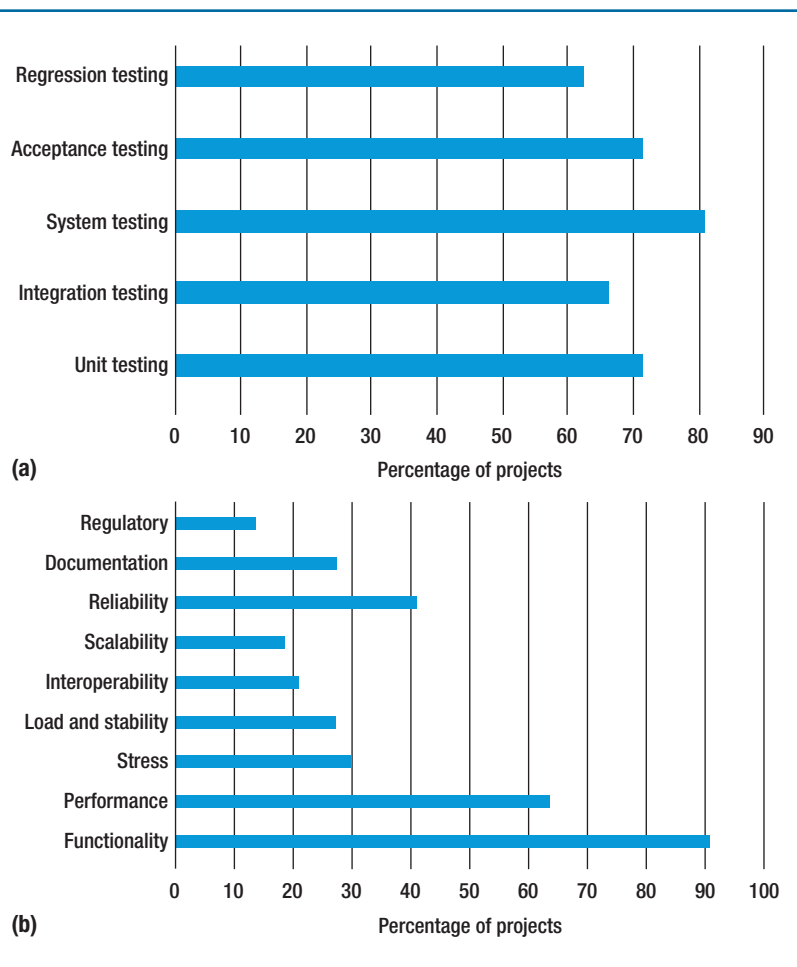


FIGURE 4. Participant responses regarding the (a) levels of testing and (b) types of system tests for their projects (122 responses total). The participants primarily used black-box techniques.

identifies test cases to reduce the number of needed test cases without reducing test efficacy—for example, by using orthogonal tests or some form of dataflow testing. Systematic test data definition identifies inputs to individual test cases through some process that isn't purely random.

Although 66 percent of the participants preferred to write test cases before writing code, only 26 percent actually did this. Surprisingly, the current practice of writing test cases before writing code was more

popular for the participants involved with non-safety-critical systems than for those involved with safety-critical systems. However, although the former group seemed willing to improve the situation even more, the latter showed no interest in changing toward a more test-driven development. This is consistent with Haberl and his colleagues' results. This is noteworthy, considering that empirical studies seem to ascribe high external quality to test-driven-developed code.¹

An interesting corollary to test case writing is the test case representation's degree of formality. Approximately 38 percent of the participants expressed test cases freely in verbal or text-based forms. Thirty-seven percent used formal languages, showing that this approach is becoming established. The participants used both open source and proprietary tools to write test cases. They used open source tools (for example, Junit) mostly for unit testing and proprietary tools (for example, Jira and HP Quality Center) mostly for higher-level testing. This is consistent with Haberl and his colleagues' results.¹

Almost 47 percent of the participants reported highly effective test cases—that is, the test cases discovered most or all defects. Thirty-seven percent reported medium effectiveness, and 11 percent reported low effectiveness.

Sixty-nine percent of the participants affirmed that the expected test results were available in the test cases before test execution. This survey result is a positive surprise. However, approximately 66 percent manually compared the expected and actual results. Only 34 percent stated that such comparison was normally automated and was performed with tools.


Because organizations must take data protection seriously, it was disappointing that only 22 percent of the participants tested with original data from production (see Figure 5a). Twenty-eight percent used an (anonymized) copy of the production data, and only 22 percent comprehensively documented the test data. In addition, 62 percent didn't explicitly distinguish between test case generation and generation of the associated test data.

Most participants used more than one testing environment. Approximately 76 percent used the development system as a testing environment; 39 percent even used the production system as a testing environment (see Figure 5b). Only 45 percent used a separate test system.

Metrics and Defect Management

The most-used metric for test control was requirements coverage (66 percent), followed by test case execution rate (60 percent). Although you could infer a high level of test process maturity from this, 39 percent of the participants ended their test activities when they reached the delivery deadline, and 9 percent ended test activities when the test budget was exhausted (see Figure 6).

Seventy-two percent of the participants reported that the discovered defects were related to requirements problems such as omissions, changes, and errors (see Figure 7). The next-most-reported cause was design problems (66 percent). Jira was the most popular defect-reporting tool (25 percent of all tool use).

We hope our survey and results stimulate research into the prevailing software practices. Moreover, we intend these results to highlight the areas of software testing that need the attention of both the research community and industry professionals. 

References

1. P. Haberl et al., *Survey 2011: Software Test in Practice*, German Testing Board, May 2011; www.istqb.org/documents/Survey_GTB.pdf.

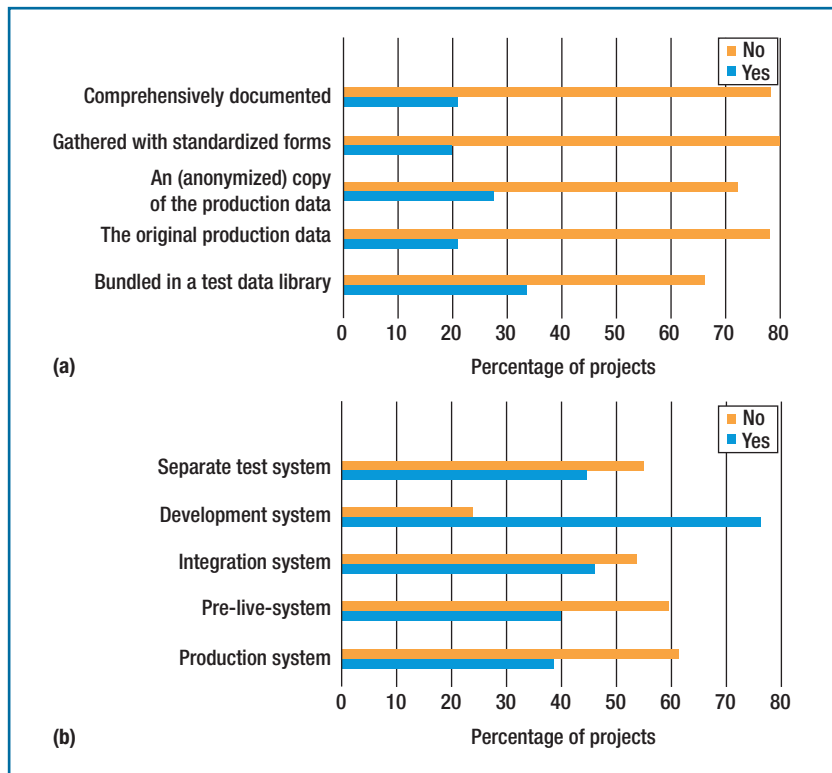


FIGURE 5. Participant responses regarding (a) test data usage (113 responses) and (b) testing environments (112 responses). Disappointingly, only 22 percent of the participants tested with original data from production. Most participants used more than one testing environment.

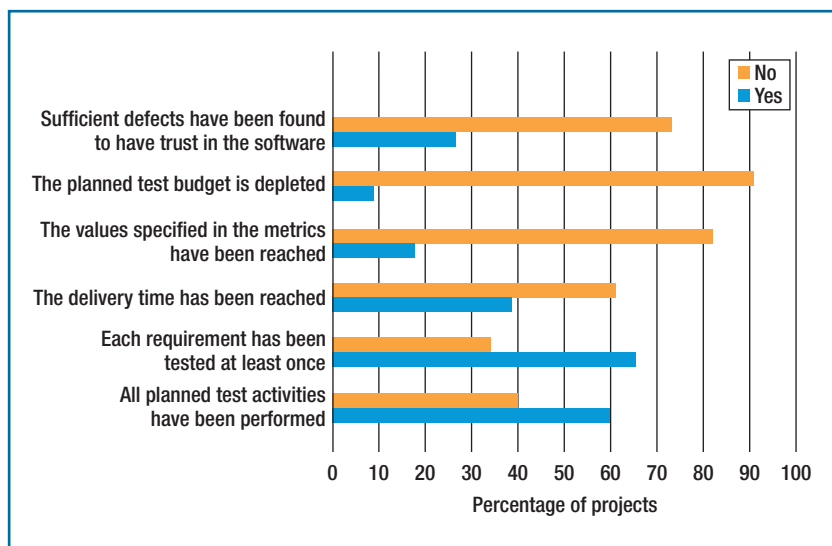


FIGURE 6. Participant responses regarding the exit criteria for test completion (103 responses). The most common criterion was to test each requirement at least once.

ABOUT THE AUTHORS



MOHAMAD KASSAB is an assistant professor of software engineering at Penn State Great Valley. His research interests include developing formal, integrated, and quantitative approaches; architectural frameworks; and tools for modeling and assessing software quality requirements. Kassab received a PhD in computer science from Concordia University, Montreal. Contact him at muk36@psu.edu.



JOANNA F. DEFRANCO is an assistant professor of software engineering at Penn State Great Valley. Her research interests include software engineering teams, project management, and effective collaboration. DeFranco earned a PhD in computer and information science from the New Jersey Institute of Technology. Contact her at jfd104@psu.edu.



PHILLIP A. LAPLANTE is a professor of software and systems engineering at Penn State Great Valley. His research interests include real-time systems, real-time image processing, safety-critical software systems, and software quality. Laplante received a PhD in computer science from the Stevens Institute of Technology. Contact him at plaplante@psu.edu.

2. S.P. Ng et al., "A Preliminary Survey on Software Testing Practices in Australia," *Proc. 2004 Australian Software Eng. Conf.*, 2004, pp. 116–125.
3. A. Causevic, D. Sundmark, and S. Punnekkat, "An Industrial Survey on Contemporary Aspects of Software Testing," *Proc. 3rd Int'l Conf. Software Testing*, 2010, pp. 393–401.
4. *Turkey Software Quality Report 2014–2015*, Turkish Testing Board, 2014; www.istqb.org/documents/TurkeySoftwareQualityReport_2014_2015.pdf.
5. *ISTQB Effectiveness Survey 2013–14*, Int'l Software Testing Qualifications Board, 2014; www.istqb.org/documents/ISTQB_Effectiveness_Survey_2013_14.pdf.
6. K. Naik and P. Tripathy, *Software Testing and Quality Assurance: Theory and Practice*, 1st ed., John Wiley & Sons, 2008.

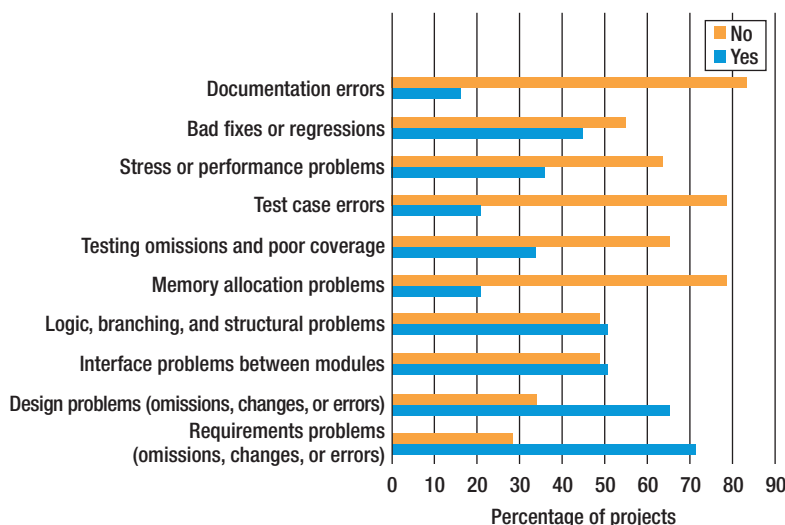


FIGURE 7. Participant responses regarding the causes of discovered defects (98 responses). The most reported cause was requirements problems.

myCS Read your subscriptions through the myCS publications portal at <http://mycs.computer.org>