A Survey of Unit Testing Practices

Per Runeson, *Lund University*

Companies participated in a survey to define unit testing and evaluate their strengths and weaknesses at applying it. Others can use the survey to judge and improve their own practices.

"Unit testing is testing of individual units or groups of related units."[1] You know the definition by the book, but what does it mean to you? What are a company's typical strengths and weaknesses when applying unit testing? Per Beremark and I surveyed unit testing practices on the basis of focus group discussions in a software process improvement network (SPIN) and launched a questionnaire to validate the results. I aimed to go beyond standard terminology definitions and investigate what practitioners refer to when they talk about unit testing. Based on this common understanding, I also investigated unit testing practices' strengths and weaknesses.

The survey revealed a consistent view of unit testing's scope, but participants didn't agree on whether the test environment is an isolated harness or a partial software system. Furthermore, unit testing is clearly a developer issue, both practically and strategically. Neither test management nor quality management seem to impact unit testing strategies or practices. Unit tests are structural, or white-box based, but developers rarely measure their completeness in terms of structural coverage. Most of the companies I surveyed desired unit test automation but had trouble spreading good practices across companies.

This survey is an indication of unit testing in several companies. You can use the questionnaire at your own company to clarify what you mean by unit testing, to identify the strengths and weaknesses of your unit testing practices, and to compare with other organizations to improve those practices.

The survey

SPIN-syd is a noncommercial network focused on software process improvement issues. It comprises representatives from 50 companies with software as a major part of their business. The companies range from consultancy firms with one employee to regional branches of multinational companies with hundreds of developers. The network represents various application domains with a focus on embedded systems. Lund University researchers and PhD students also belong to the network. The network has a monthly three-hour meeting and occasionally launches working groups around specific themes. Typically, 10 to 15 companies are represented at each monthly meeting, depending on the topic and the companies' workloads.

We conducted the survey at two meetings. During the first meeting, we held a focus group discussion about unit testing. Participants included 17 representatives from 12 companies, the moderator (a software quality manager), and the secretary (me).

Table 1 lists the participating companies' characteristics.

Eight representatives from seven of those companies took part in the subsequent survey, along with representatives from seven new ones. The participating companies represent automation, banking, case tools, information systems, health care, transportation, and telecom. Most of the consulting companies have their primary occupation in telecom. The participants, who range from testers to quality managers, are interested in testing and software quality issues in general. Of particular interest for unit testing practices, one company is developing safety-critical software, and another uses agile development methods.[2]

Figure 1 outlines the procedures for conducting the survey:

1. The participants each spent a few minutes reflecting on a question and taking notes on a chart.
2. Each participant presented his or her view of the question, and the group discussed the results with the moderator's input.
3. After the meeting, I documented the findings and fed them back to the participants for review.
4. I later qualitatively analyzed the findings using Zachman's framework (see the sidebar). The results constitute this article's core.

We initiated the focus group discussions around three themes:

- What is unit testing?
- What are the participants' strengths regarding unit testing?
- What are the participants' problems regarding unit testing?

Three steps comprised the second phase

5. I prepared a survey questionnaire on the basis of the focus group discussions.
6. Participants filled out the questionnaire in a SPIN monthly meeting or via email, with the main purpose of validation (that is, determining whether the finding is unique to a single company or widespread in the community).
7. Finally, I analyzed the questionnaire responses, primarily using descriptive statistics.

The questionnaire consisted of 26 questions on what unit testing is and 24 questions on its strengths and weaknesses in the respondent's organization (see the sidebar "Questionnaire Instrument" on page 25). Answers were given on a five-level Likert scale—that is, on a scale from "strongly agree" to "strongly disagree" and "very good" to "very bad." All questions had a "not applicable" option.

The questionnaire responses indicate agreement between respondents. The analysis compares the focus group results and the questionnaire results. Given the number of respondents, I couldn't compare individual companies or application domains.

Table 1

Companies represented in the focus group meeting and the questionnaire

Company	Application domain	Size*	Participants in focus group	Participants in questionnaire
1	Telecom	Large	3	1
2	Automation	Medium	3	2
3	Case tools	Small	2	1
4	Information systems	Large	1	1
5	Banking	Medium	1	1
6	Consulting	Medium	1	0
7	Health care	Medium	1	1
8	Health care	Medium	1	0
9	Consulting	Small	1	0
10	Information systems	Small	1	0
11	Consulting	Extra small	1	0
12	Consulting	Extra small	1	1
13	Case tools	Large	0	1
14	Telecom	Large	0	1
15	Banking	Medium	0	1
16	Consulting	Medium	0	1
17	Consulting	Medium	0	1
18	Telecom	Medium	0	1
19	Transportation	Medium	0	1
Total			17	15

**Number of developers in the surveyed company: extra small is 1, small is 2–9, medium is 10–99, and large 100–999.*

What is unit testing?

Tim Koomen and Martin Pol define a unit test as "a test, executed by the developer in a laboratory environment, that should demonstrate that the program meets the requirements set in the design specification."[3] James Whittaker states that "unit testing tests indi-

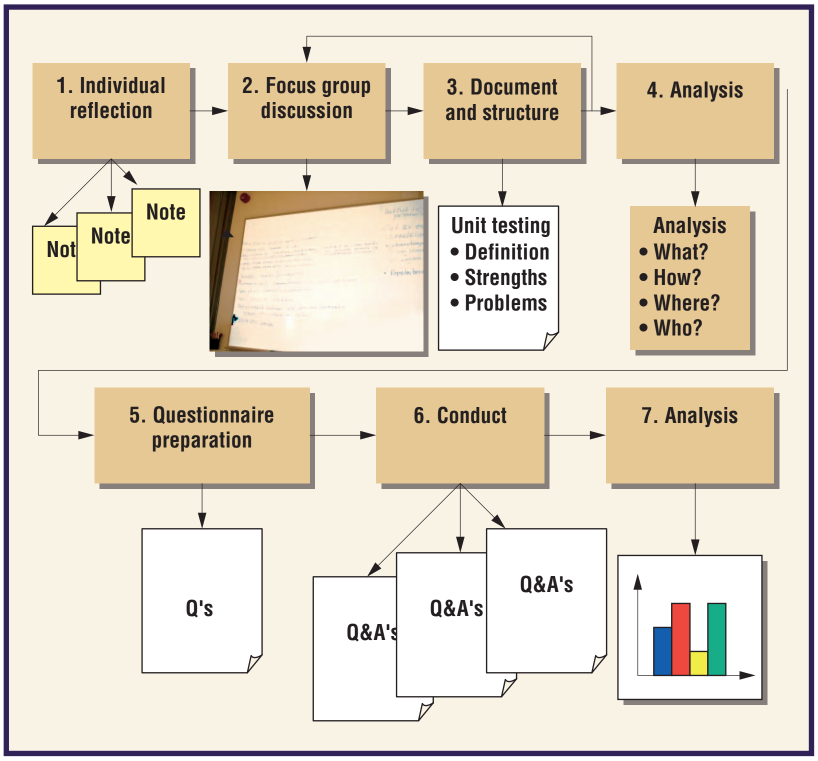

Figure 1. Survey methodology overview.

vidual software components or a collection of components. Testers define the input domain for the units in question and ignore the rest of the system. Unit testing sometimes requires the construction of throwaway driver code and stubs and is often performed in a debugger."[4]

Because verbal definitions of unit testing already exist, I didn't restrain the discussion as such. Instead, I aimed for a broader understanding of what *unit test* means and what role it plays in an organization. I report the results, structured according to Zachman's framework, except that no observations were made regarding the location dimension (*where*). Figure 2 (page 26) summarizes the questionnaire responses. For each item, I first report the focus group discussions and then the questionnaire results. (Throughout this article, Q*n*.*m* refers to question number *m* in Figure *n*.)

What?

Unit testing means testing the smallest separate module in the system. Some people (such as Koomen and Pol[3]) stress that it's the smallest *specified* module, but opinions differ about the need for specifications. Regardless, unit testing is technically oriented, with in/out parameters.

Nothing from the focus group discussions contradicted this definition. The only variation was whether developers should specify modules and tests. Nonetheless, the unit testing practice is sometimes different.

The questionnaire confirmed that respondents considered unit tests to be technical tests focused on the system's smallest units (Q2.1–3). Many disagreed whether they should execute the unit test in a scaffolding environment or conduct it in an almost-complete system environment (Q2.4). But according to Whittaker, unit testing should "ignore the rest of the system."[4] So, it can be run in the complete system environment, focusing on the unit under test and ignoring the rest.

The focus group attendants who also responded to the questionnaire agreed on the definitions to a larger extent than the other respondents.

How?

From the focus group discussions, I found that companies conduct unit testing on the basis of the program's structure (that is, white-box or grey-box testing). They want the test cases to be repeatable and also automated with respect to test execution and result checking. They can conduct unit testing in the form of test-driven design.[5]

The questionnaire indicated that structural

Zachman's Framework

John Zachman presented a framework for analyzing information systems architectures.[1] The framework has six categories—what, how, where, who, when, and why—although these terms weren't originally used. For each category, questions are defined and tailored to the domain under investigation. Although Zachman originally intended the framework for information systems development, he proposed that it might also be used for creating new approaches to system development.

Joseph Feller and Brian Fitzgerald used the framework to analyze open source development,[2] and Peter Greberg and I used the same principles to analyze Extreme Programming and the Rational Unified Process.[3]

In the main article, I use Zachman's framework to structure the outcome of the focus group meetings and to define the validation questionnaire.

References

1. J.A. Zachman, "A Framework for Information Systems Architecture," *IBM Systems J.*, vol. 26, no. 3, 1987, pp. 276–292.
2. J. Feller and B. Fitzgerald, "A Framework Analysis of the Open Source Development Paradigm," *Proc. 21st Int'l Conf. Information Systems,* ACM Press, 2000, pp. 58–69.
3. P. Runeson and P. Greberg, "Extreme Programming and Rational Unified Process—Contrasts or Synonyms?" *Experience Session Proc. European Software Process Improvement and Innovation Conf.* (EuroSPI 05), John von Neumann Computer Soc., 2005, pp. 1.1–7.

Questionnaire Instrument

For each question, respondents chose the best answer from the following scale: "strongly agree," "agree," "neutral," "disagree," "strongly disagree," or "not applicable."

To what extent do you agree with the statements below?

What is a unit test?

1. A test of the smallest separate unit
2. A technical test with in/out parameters
3. A test that focuses on separate functions
4. A test that must execute separately from rest of the system

How is a unit test conducted?

5. Based on the program structure
6. Monitored by coverage measurements
7. Automated execution
8. Automated follow-up

How do developers conduct the unit tests?

9. Specified in text
10. Specified in test code
11. Executed by developers themselves

Who decides how the unit test shall be conducted?

12. Developers/testers
13. The test department
14. The quality department

When are the unit tests executed?

15. For each compilation/system build
16. Many times daily
17. At least daily
18. At least weekly

How much time does it take to run all unit tests?

19. Seconds
20. Minutes
21. Hours

Why are unit tests conducted?

22. To ensure that the unit functions as expected
23. To accept a unit from other sources
24. To specify a unit (test first)
25. To improve the product quality in general
26. To meet customer requirements

For each question, respondents chose the best answer from the following scale: "very good," "good," "neutral," "bad," "very bad," or "not applicable."

How does the following function in your organization?

What?

1. Identify suitable units
2. Test GUI components
3. Test data-dependent functions
4. Test depending on external code
5. Test real-time aspects
6. Test case selection
7. Maintenance of test code

How?

8. Specify good unit test cases
9. Automate unit testing
10. Build/tailor frameworks for unit testing
11. Integration with build systems
12. Integration with configuration management systems
13. Integration to trouble-reporting systems
14. Unit test documentation
15. Scaffolding (stubs and drivers)
16. Coverage or other test progress measurements

Who?

17. Developers execute all needed unit tests
18. Unit testing external units is sufficient
19. Testing has sufficient priority and status
20. Developers have sufficient competence and skills to conduct unit testing

When?

21. I execute the unit test frequently enough
22. I have clear criteria to judge when the unit test is finished

Why?

23. The developers are motivated to execute unit tests
24. I know that I gain more than the cost for time spent on unit testing

basis is important (Q2.5), although it doesn't have to be formally measured through coverage measures (Q2.6). Regarding automation, the automation's execution is more important than automatic result checks of test cases (Q2.7–8). Furthermore, unit tests are documented in test code rather than in text (Q2.9–10).

Who?

The focus group agreed that developers and development teams conduct unit tests. How-

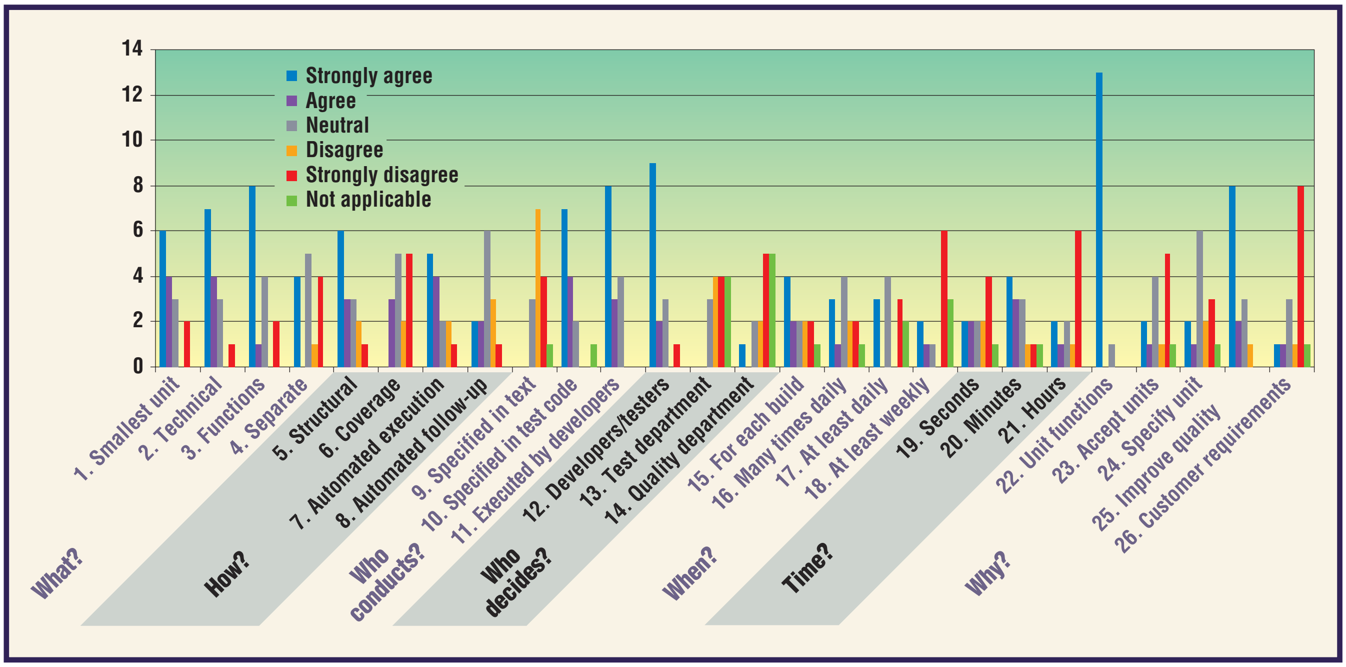

Figure 2. Response frequencies on the questionnaire regarding unit test definitions (Q2.1–26).

ever, they disagreed whether unit tests must be specified.

The questionnaire confirms that the unit tests are the development organization's concern (Q2.11–12). Neither the test organization nor the quality organization has any say (Q2.13–14). (Similarly, Koomen and Pol state explicitly that the developers execute the unit test,[3] but this is implicit in other definitions.)

When?

The focus group mentioned that unit tests give developers quick feedback. The group didn't discuss time in relation to project phases, although unit tests are implicitly connected to the implementation activities.

How often each company executed the unit tests varied widely (Q2.15–18), as did the execution time. Most respondents said that running all unit tests took just seconds or minutes, but some respondents had unit test suites that took hours to execute (Q2.19–21).

Why?

The focus group stated that unit testing's main focus is assuring the system's functionality. Unit testing doesn't consider any extra-functional aspects because it runs separately from the system. For those executing unit tests in a complete system environment, the distinction isn't as clear. The group stressed that testing verifies that a module has the functionality a developer expects, which isn't necessarily what other stakeholders expect. In test-driven design, unit tests define the problem and can contribute to less complex solutions when applying refactoring.

The questionnaire confirms unit testing's functional focus (Q2.22). Unit testing's purpose is related to general quality improvements (Q2.25). In a few cases, companies use unit testing for internal acceptance (Q2.23) or as a technical specification (Q2.24). In very few cases do customers explicitly require unit tests (Q2.26).

Unit testing strengths

The focus group members discussed their strengths with regard to unit testing. The conversation was an honest sharing of good practices that other noncompetitive companies could use. SPIN-syd's tradition of openness between peers as well as toward researchers and other external sources[6] reduces the risk of people telling success stories without a solid foundation.

I report the unit testing strengths according to Zachman's framework. For the questionnaire and the responses, see the sidebar "Questionnaire Instrument" and figure 3, respectively. This section discusses the positive answers, and the next section discusses the negative answers (Q3.12, 15, 17, 18). Real-time issues weren't applicable for many respondents (Q3.5).

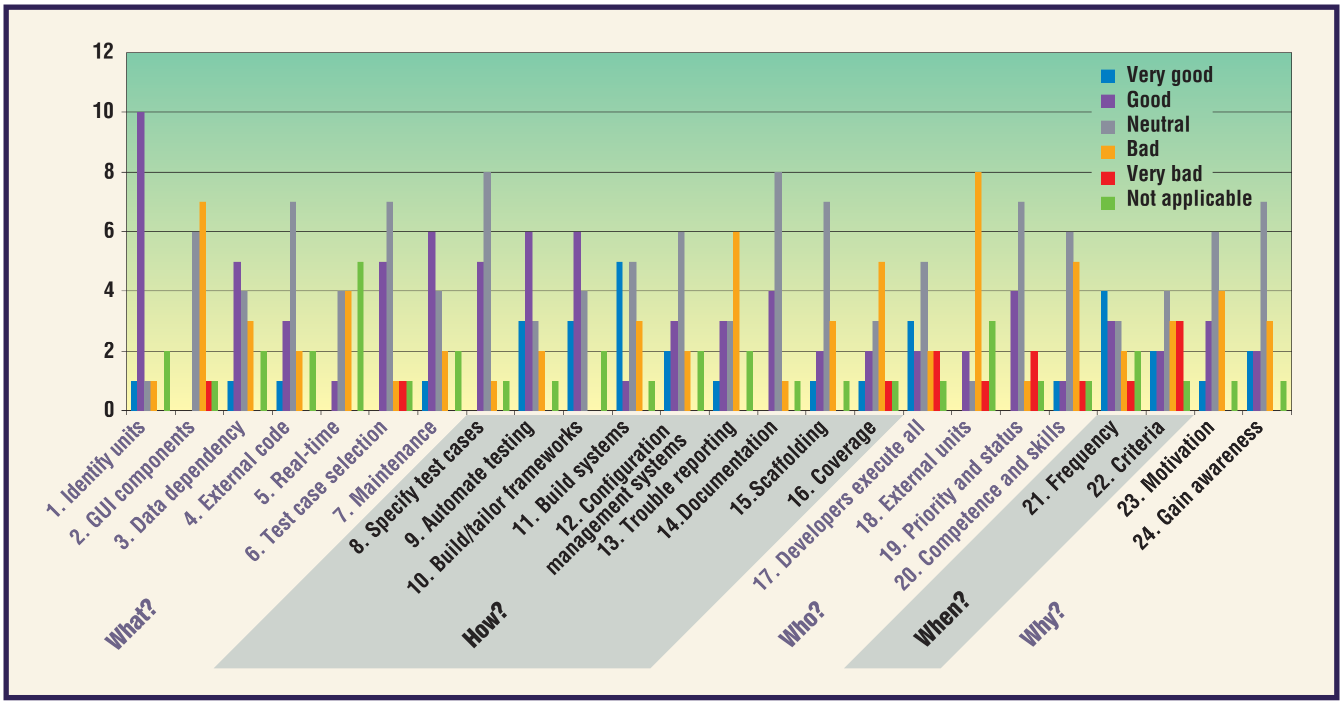

Figure 3. Response frequencies on questionnaire regarding unit test strengths and weaknesses (Q3.1–24).

What?

The focus group participants considered it a successful practice to unit test modules other than the one under test. One company applies test-driven design and has an automated unit test suite that clearly reveals changes in other modules. This indicates that unit tests are conducted in an environment where other system modules exist.

The questionnaire reveals that using unit testing for external modules wasn't common practice but rather a single example (Q3.4). Furthermore, most organizations can easily identify units (Q3.1) and maintain unit test code (Q3.7).

How?

Two companies in the focus group considered it a strength that they have set up a framework for unit test automation. If this framework has internal support, it will improve the practice. Furthermore, unit testing should integrate with the build system—that is, companies should automatically run a selected set of test cases for every version of the system.

The questionnaire confirms that test automation and tailoring frameworks for unit testing are successful practices (Q3.9–10). The build system was judged neutral to very good (Q3.11), and the specifications were considered neutral or good (Q3.8).

Who?

Although the focus group defined unit testing as conducted by developers and development teams, they also favored independent or third-party unit tests. Companies could achieve this by widening the unit testing scope somewhat—that is, enlarging the group of modules under test and then allowing testing by both their own development team and other teams.

Regarding testing competence, companies considered the SPIN-syd forum for cross-company learning valuable. However, the questionnaire didn't present competency as a strength. The status and priority given to unit testing were judged neutral (Q3.19).

When?

Continuous regression tests are a strength. One company runs automatic tests every night, including automatic results checking. Another company runs regression tests when refactoring the code. It also runs memory tests continuously, ensuring that the basic functionality and characteristics are in place.

This practice isn't widespread. The questionnaire shows that regarding unit test execution's frequency, the answers range evenly over the scale (Q3.21).

Why?

For single companies in the focus group,

The network's tradition of openness is a solid basis for achieving a satisfactory picture of the practitioners' unit testing problems.

the motivation for unit tests must accompany external requirements (Q2.26). For example, safety standards forced a company developing safety-critical software to run automatic unit tests. Now, when the tests are implemented, the company is highly satisfied with them.

Another motivating factor is the use of agile methods. The test suites could function as a technical specification, which is continuously updated when the interfaces change.

Unit testing weaknesses

The focus group discussed weaknesses, and I validated them in the questionnaire. Again, the network's tradition of openness is a solid basis for achieving a satisfactory picture of the practitioners' unit testing problems.

What?

A common issue was testing GUI modules (in particular, concerning automation). Also, identifying units for test was a problem; the objects under test tended to be "groups of related units" rather than "individual units."[1]

Many companies desired test automation, but those that started automation initiatives soon accumulated large volumes of code. The companies had to maintain this code due to product evolution, requiring much effort.

Focus group members saw a unit testing problem in testing modules that have or depend on large data structures. It's hard to create a relevant test environment for modules interacting with a complex system state or a complex system environment.

The questionnaire confirmed the GUI problems (Q3.2), ranking it as the most problematic area. On the contrary, companies judged unit identification as "good" (Q3.1), while the focus group considered it unclear. The questionnaire labeled test automation "good" (Q3.9) and data-dependent unit testing "neutral" (Q3.3).

How?

The focus group named documentation as a problem. Without proper documentation, repetitiveness is only informal. One proposed solution is test automation, adding incentives for keeping the test scripts up-to-date, but (of course) at a cost.

Test frameworks were considered a key to success, but they have integration problems. Like any new software engineering approach, it's easier to get it to work when beginning from scratch, which is rare because most projects build on legacy code.

The focus group considered test case selection for a continuous regression test a hard task. The companies used no systematic approach to test selection but relied on the developers' expertise and judgment, which might mean executing too many or too few test cases.

Test metrics are scarce, so quantitative methods weren't used for test management to a large extent. (An earlier survey also found this to be true.[6])

The questionnaire responses were quite neutral regarding documentation (Q3.14). Regarding test case selection, the respondents were neutral (Q3.6) but agreed that the state of coverage and other progress measures were "bad" (Q3.16). The same holds for trouble reporting (Q3.13).

Who?

The focus group considered unit testers' competency very important. Combined with a low-status attitude toward testers, it's hard to find the person with the right knowledge and skill. By definition, developers themselves conduct the unit tests. They know the modules very well—perhaps too well to set up a set of critical tests.

Few companies had a strategy for unit testing. Instead, the developers themselves defined the ambitions for unit testing, leading to varying practices.

Responses were neutral to whether unit testing competency is sufficient (Q3.20). Developer motivation needs improvement (Q3.23).

When?

When do you know a module has been sufficiently tested? Some focus group members favored coverage criteria, but others advocated giving priority to test cases' quality (that is, their ability to reveal faults).

The questionnaire responses showed no clear tendency toward a specific stopping criterion, although there was a slight balance toward lack of well defined and monitored criteria (Q3.16, 22).

Why?

How much time should companies spend on unit testing? How much should they devote to automating unit testing? The focus group discussed the trade-off between cost and gains, concluding that it's hard to judge. Test managers need arguments and models to calculate the return on the test investment, but no such models exist.

The questionnaire makes clear that it's difficult to motivate the developers to execute unit tests (Q3.23). Also, there's no explicit return on investment calculation on whether the gains outweigh the costs of unit testing (Q3.24).

The survey's use

Table 2 summarizes the survey's results. How can we use these results? Are they valid outside the surveyed organizations?

The respondents represent companies of different sizes and application domains, making the results more externally valid (meaning that the findings are valid outside the limited set of surveyed companies). However, we should take the results not as a universal view of unit testing's status but as a starting point for a universal discussion and analysis. We should use the survey to help define what a company means by "unit testing," as a benchmark for judging a company's test performance, and as a starting point for an improvement initiative.

A company with an unclear definition of unit testing runs the substantial risk of bad or inconsistent testing. With a clear and shared understanding of unit testing in a specific environment, different stakeholders will likely understand and accept their responsibilities. The first part of the questionnaire can help support this goal.

When a company shares an understanding of the definitions, the people can discuss the unit testing practices' strengths and weaknesses. It's important to start small in an improvement program to gain sustainable improvements over time.[3,7] The second part of the questionnaire can support a lightweight assessment to identify improvement needs. It can also serve as a basis for comparing different noncompetitive companies—a feature successfully used in SPIN-syd.

Acknowledgments

I thank Per Beremark for initiating and moderating the focus group within SPIN-syd. Thanks to all participants in the SPIN-syd network for contributing to the survey and to Carina Anderson for reviewing this article. The Swedish Research Council funded this research under grant 622-2004-552 for a senior researcher position in software engineering.

Table 2

Unit test definitions, strengths, and problems according to the survey

	Definitions	Strengths	Problems
What?	Test of smallest unit or units	Unit identification Test of surrounding modules	GUI test Unit identification Test scripts and harness maintenance Data structures
How?	Structure-based Preferably automated	Test framework	Documentation Framework tailoring Test selection Test metrics
Where?	Solution domain	None found	None found
Who?	By developer	Independent test Competence network	Competency Independence Introduction strategy
When?	Quick feedback	Continuous regression test	Stopping criteria
Why?	Ensure functionality	External requirement (safety) Agile methods	Cost versus value

About the Author

Per Runeson is a professor of software engineering at Lund University and the leader of the Software Engineering Research Group. He is also a Swedish Research Council–funded senior researcher. His research interests include software development methods and processes, particularly verification and validation methods. He received his PhD in software engineering from Lund University. He's an editorial board member of the *Empirical Software Engineering Journal* and *Journal of the Association of Software Testing.* He's a senior member of the IEEE. Contact him at the Dept. of Communication Systems, Lund Univ., Box 118, SE-22100 Lund, Sweden; per.runeson@telecom.lth.se.

References

1. *IEEE Std. 610.12-1990, Standard Glossary of Software Engineering Terminology*, IEEE, 1990.
2. D. Karlström and P. Runeson, "Combining Agile Methods with Stage-Gate Project Management," *IEEE Software*, vol. 22, no. 3, 2005, pp. 43–49.
3. T. Koomen and M. Pol, *Test Process Improvement—A Practical Step-by-Step Guide to Structured Testing*, Addison-Wesley, 1999.
4. J.A. Whittaker, "What is Software Testing? And Why Is It So Hard?" *IEEE Software*, vol. 17, no. 1, 2000, pp. 70–79.
5. K. Beck, *Test Driven Development: By Example*, Addison-Wesley, 2003.
6. P. Runeson, C. Andersson, and M. Höst, "Test Processes in Software Product Evolution—A Qualitative Survey on the State of Practice," *J. Software Maintenance and Evolution*, vol. 15, no. 1, 2003, pp. 41–59.
7. D. Karlström, P. Runeson, and S. Nordén, "A Minimal Test Practice Framework for Emerging Software Organisations," *Software Testing, Verification and Reliability*, vol. 15, no. 3, 2005, pp. 145–166.