

On the Testing Maturity of Software Producing Organizations

Mats Grindal
Humanities and Informatics
University of Skövde, Sweden
magr@enea.se

Jeff Offutt
Info. and Software Engng
George Mason University
Fairfax, VA 22030, USA
offutt@ise.gmu.edu

Jonas Mellin
Humanities and Informatics
University of Skövde, Sweden
jonas.mellin@his.se

Abstract

This paper presents data from a study of the current state of practice of software testing. Test managers from twelve different software organizations were interviewed. The interviews focused on the amount of resources spent on testing, how the testing is conducted, and the knowledge of the personnel in the test organizations.

The data indicate that the overall test maturity is low. Test managers are aware of this but have trouble improving. One problem is that the organizations are commercially successful, suggesting that products must already be “good enough.” Also, the current lack of structured testing in practice makes it difficult to quantify the current level of maturity and thereby articulate the potential gain from increasing testing maturity to upper management and developers.

1 Introduction

Studies from the 1970s and 1980s claimed that testing in industry consumes a large amount of resources in a development project, sometimes more than 50% [4, 7, 10, 20]. A recent study found that, at least for some distributed systems, there has been a significant shift of the main development cost from programming to integration and testing [5].

There has also been a steady increase in the quality requirements of software, partly led by the increasing emphasis on application areas that have very high quality requirements, such as web applications and embedded software [18].

The high cost of testing and the trend toward increased focus on the quality of software should be strong incentives for software development organizations to improve their testing. However, our experience from industry is that the test maturity of many organizations is still low. Further, it is our perception that even though there is a great need for improving the quality of software testing, lots of techniques

have been developed, and numerous commercial tools are available, most organizations do not make frequent or effective use of the tools.

This paper presents data from a documentation and assessment of the test maturity of twelve software producing organizations. The main purpose of this study is to provide industry and academia with a starting point for discussions on how to improve. The test maturity of an organization depends on many factors. For instance, the Test Process Improvement (TPI) method [15], defines 20 different areas that taken together describe the test maturity.

In this study, we are interested in aspects of test maturity that relate to the use of methods for selecting test cases. The reason for this narrowed scope is that an abundance of test case selection methods have existed for a long time [16, 3], but are rarely used in industry. This study also reasons about the factors that influence the application of testing research results in industry.

Other aspects of test maturity are equally valid to explore. For instance, Runesson et al. [19] examine how test processes are defined and used in practice.

An early decision of this study was to focus on a diverse set of organizations instead of one type. A diverse sample makes it possible to compare groups of organizations, which may help identify patterns that can be further explored in future studies. With diversity, the results should also appeal to a larger audience. The down-side is that it is harder to draw general conclusions from a diverse set. The twelve organizations investigated were selected to be diverse in terms of age, size, type of product produced and how long the development projects usually last.

In the scope of this paper, the term *testing* is used in a wide sense. It includes pre-execution testing such as reviews of requirements and validation through prototyping as well as all test case execution activities. The main reason for this is our interest in the use of test strategies as a way to coordinate all of the verification and validation activities. Most organizations in our sample used the term *testing* in this way. The more refined term of *test case selection* is

used to mean a specific procedure for selecting values for tests.

Section 2 describes how this study was performed, including how the organizations investigated were selected, how the data were collected, and how the data analysis was done. This section also discusses aspects of validity with respect to this study. Section 3 presents our collected data and section 4 analyzes these data and discusses the results. Section 5 concludes this study with a short summary.

2 The Study

This test maturity study was performed as a series of interviews with representatives from twelve organizations. It is a qualitative study with some quantitative elements. The forthcoming sections describe how this study was carried out in more detail.

2.1 Research Questions

This study had six distinct research questions, the primary one being (Q1:) Which **test case selection** methods are used in the development projects? Some additional research questions were also used to allow for deeper analysis of the results. These questions are (Q2:) Is the testing in the development projects guided by a **test strategy**? (Q3:) When are **testers** first involved in the project? (Q4:) What is the **general knowledge** of the testers? (Q5:) How much of the **project resources** are spent on testing? (Q6:) Which **metrics** are collected and used during testing?

To determine the diversity of the sample, data on several organizational properties, such as age, size, types of product developed, etc. were also gathered.

2.2 Organizations Investigated

The subjects of this study were customers of Enea Test AB, the first author's employer. Enea Test AB provides testing and test education consultancy services. A list of organizations was assembled to achieve a spread across age, size and type of products developed. The organizations on the list were contacted and asked if they were willing to participate in the study. They were also asked to confirm our preliminary estimates of their organizational properties.

Thirteen organizations were contacted, and one declined to participate. The contacts at the remaining 12 organizations were managers with testing as part of their responsibility.

Figures 1 through 6 show the spread across age and time since the last major reorganization, size of company, size of development organization, size of normal projects, and type of product developed. Figure 1 shows that the organizations

range in age from three to fifty years, and the time since the last major reorganization ranges from one to eight years.

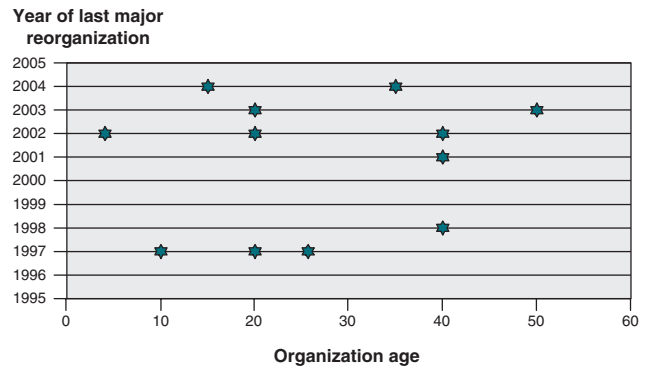


Figure 1. Age and year of last reorganization.

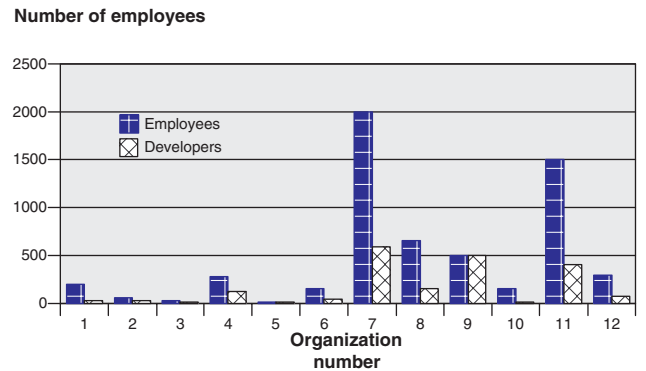


Figure 2. Number of employees & developers.

As summarized in figure 2, the size of the organizations range from 15 to 2000 employees. The size of the development departments range from 15 to 600. Six organizations have all their development concentrated at a single site, while the others are spread between two and six sites.

Figures 3 and 4 show the sizes of the projects in calendar-time and person-hours. The shortest projects take three to four months to complete while the longest projects take up to fifty months. The cost for these projects measured in person-hours range from 1,700 to 288,000 hours. When the project lengths varied within an organization, they were asked to report on their "typical" projects. Organization 10 has two types of projects, one type with very little new functionality (10a) and one with mostly new functionality (10b), and in some cases gave data for each type of project.

The organizations investigated also exhibit great variance in the number and types of products they develop. Figure 5 shows that six organizations develop embedded prod-

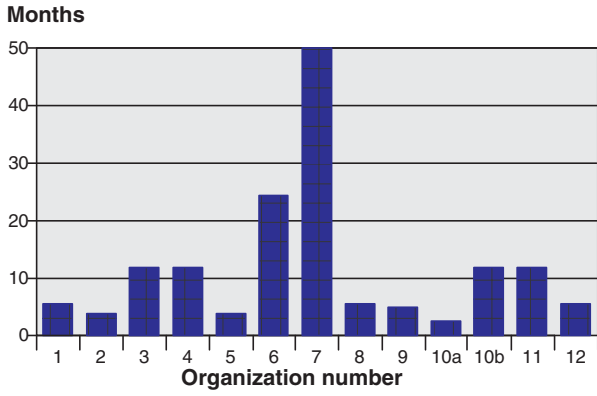


Figure 3. Size of projects (calendar time).

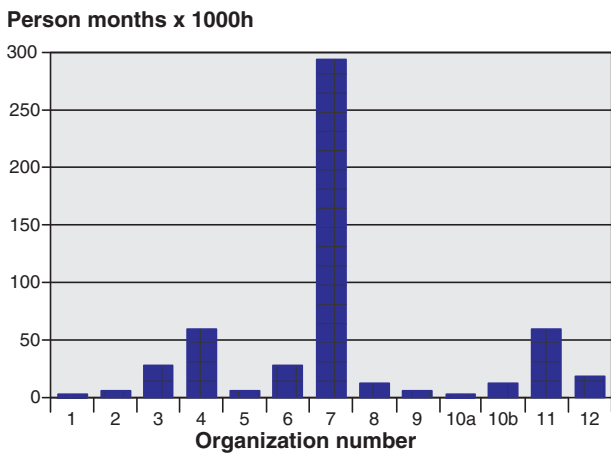


Figure 4. Size of projects (person time).

ucts, four of which are also safety-critical. The other six develop software for non-embedded systems. Figure 6 shows that all companies develop more than one product or product version at the same time. In some cases the amount of parallel development is limited to two or three products or versions of products, whereas in other cases as many as one hundred custom-designed product versions are developed simultaneously.

Taken together, the twelve organizations exhibit a wide spread across all of the investigated parameters, which enabled this study to sample from diverse organizations.

2.3 Data Collection

Each interview lasted about one hour. One researcher (the first author) met with the representative at the organizations' sites. Some representatives (called *respondents* hereafter) brought additional people to the interview to help answer questions.

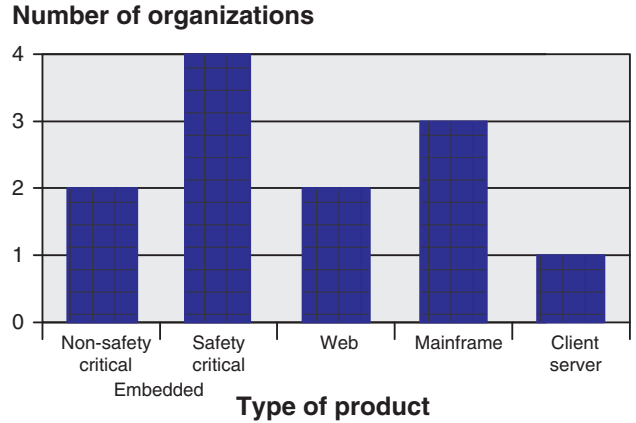


Figure 5. Number of organizations that develop each type of product.

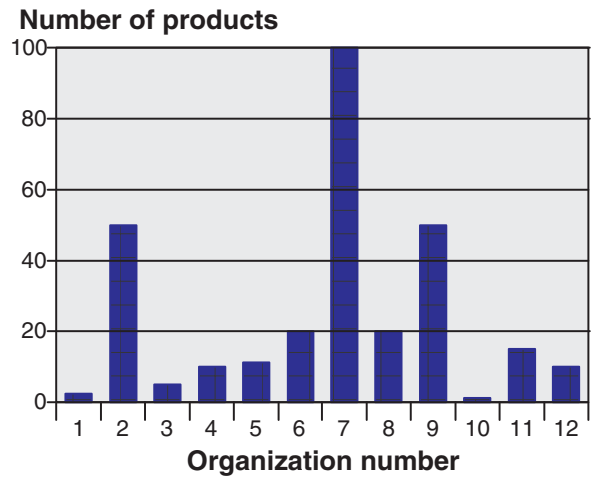


Figure 6. Number of products developed.

The respondent was given the questionnaire at the start of the interview. The interviewer and the respondent completed the questionnaire together. The interviewer guided the respondent through the questionnaire by clarifying information and helping the respondent to translate his/her vocabulary to the vocabulary used on the questionnaire. When both parties agreed to an answer, the interviewer recorded the answer in the questionnaire, in view of the respondent.

2.4 Analysis

All results were transferred into a spreadsheet and the researchers discussed the data recorded and how to present them.

The graphs were then used to identify differences and similarities among the organizations. Cross-property comparisons were then performed through Spearman tests [1]. The Spearman test compares two rankings based on ordinal values and determines the level of correlation between the two rankings. Table 1 shows the recommended interpretations of value of the Spearman coefficient. Results are documented and explained in section 4.

Values	Interpretation
$0 \leq x \leq 0.33$	Weak relationships
$0.33 < x \leq 0.66$	Medium strength relationships
$0.66 < x \leq 1$	Strong relationships

Table 1. Spearman coefficient interpretation.

2.5 Validity

Cook and Campbell [8] identify four different types of validity that need to be considered in studies of this type: conclusion validity, construct validity, internal validity, and external validity.

Conclusion validity concerns on what grounds conclusions are made, for instance the knowledge of the respondents and the statistical methods used. This study did not make an explicit evaluation of the respondents' knowledge, but all respondents are judged to be experienced, based on their positions in their organizations. All participating organizations were guaranteed anonymity, which adds to the confidence in the answers. To ensure that the interview was treated seriously, the organizations were offered a free training seminar in return for a complete interview.

Interviewer bias was handled in part by only choosing organizations that the researchers were unfamiliar with. A carefully reviewed questionnaire was also used to decrease the risk of interviewer bias. Further, all documented answers were agreed upon by the interviewer and the respondent.

Construct validity concerns whether or not what is believed to be measured is actually what is being measured. The main focus of this study is to find out which test case selection methods companies use. There is a possibility of managers giving answers that reflect the directives, rather than what is actually in use. However, we theorize that for new methods of working to be adopted in an organization as a whole, these need to be documented and communicated via the management. Thus, what management thinks is being used is relevant even if it does not match.

Another risk relating to construct validity is the different terminologies used by different organizations. This was handled by using terminology from Test Process Improvement (TPI) [15] and BS7925-1 [6]. Both TPI and BS7925-1 were known to most organizations in this study. Also, the

interviewer discussed terminology with the respondents to clarify misunderstandings.

Internal validity concerns matters that may affect the causality of an independent variable, without the knowledge or the researcher. Only one short (45-75 minutes) interview was held at each organization to reduce the risk of the interviewer becoming biased by getting to know the organization and its personnel.

Having only one respondent results in a risk that only part of the picture is revealed. This was partly addressed by having overlapping questions to be able to detect possible inconsistencies in the answers.

Some answers, for instance the level of knowledge of their test team, are bound to be inexact. This limits the ability to compare organizations, but this was not a primary goal of the study.

External validity concerns the generalization of the findings to other contexts and environments. It is inherently difficult to judge external validity of studies like this since it is impossible to know the size and distribution of the goal population. Hence, one can never know if a sample is representative or how large the sample needs to be for a defined level of confidence.

The approach taken in this study is to construct a sample that is heterogeneous with respect to a number of different properties like age, size, type of products etc. This approach limits the possibilities of making general claims about the software industry based on the results in this study. However, it is still possible to identify relationships and correlations among the studied organizations and use these as a basis for further studies.

Practical reasons limited the heterogeneity in that all organizations are Swedish. It seems unlikely that Swedish software companies would be substantially different from other European companies. There is a common perception that European companies emphasize reliability in software more than North American companies, but we know of no data to support that perception. It would be interesting to repeat this study elsewhere.

3 Observations and Data

The results of the interviews are presented in the same order as the questions in section 2.1. The organizations are identified only by number, and not name, so as to protect their privacy. Some data have been left out for space reasons. These can be found in the corresponding technical report [14].

3.1 Test Case Selection Methods

Only three of the twelve organizations report structured use of test case selection methods.

Organization 4 uses equivalence partitioning [16], boundary value analysis [16] and some basic combination strategies [13], for instance “each choice” [2]. Organization 8 uses boundary value analysis and cause-effect graphing [16] and some proprietary methods. Organization 10 tries to satisfy 100% requirements coverage to control the choice of test cases, which can be considered to be an informal test case selection method.

In the remaining nine organizations there is no enforcement of the use of test case selection methods. Instead it is up to individual testers and developers to select test cases. It is likely that some individuals use test case selection methods on their own, but the organization as a whole has no control of this.

It is interesting to note that two of the organizations that produce safety-critical products do not enforce the use of test case selection methods.

3.2 Test Strategy

One goal of a test strategy is to provide organizational global advice in finding the most important defects as early and cheaply as possible [15]. Thus, a test strategy describes the responsibility of each test phase in the project. Advice is also included on how to choose test methods and coverage criteria in each phase. The organizations were asked if they have a test strategy, if it is implicit or explicit, what types of information it contains, and if it is used.

Figure 7 shows that three of the twelve organizations do not use a test strategy at all. Two have explicit test strategies written but do not use them, three use implicit test strategies, mostly embedded as advice in their test processes or in some cases as part of some standard regulating the testing of certain aspects of the product. Only four use explicit test strategies.

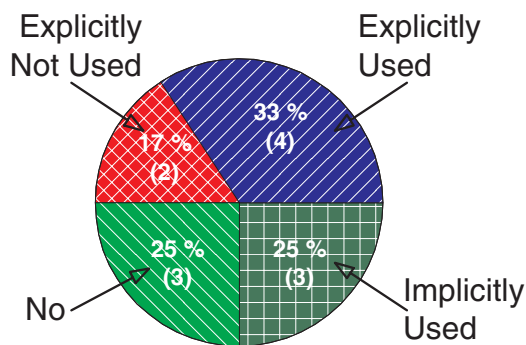


Figure 7. Does a testing strategy exist?

The type of information included in the test strategies varies. Four organizations have information about the test phases, three have pointers to standards, and only three have information about test case selection methods. These three

organizations are the same organizations that report structured use of test case selection methods.

3.3 Moment of Involvement

It is a general view in the test community that testers should be involved early in the projects [9, 11]. There are several reasons for this. One reason is to use time effectively during test execution by preparing all tests prior to the test execution. Another reason is that testers can help detect and remove faults even before implementation.

Six organizations involve their testers at the start of the project. Another four involve their testers during requirements collection. When testers are involved in projects early, their main task is usually test case design, which may lead to improved software requirements. In some cases the testers also participate in requirements review.

The final two organizations do not involve their testers until the product is ready to be delivered to the test organization.

3.4 Test Team Knowledge

One possible reason why structured testing is not used by organizations is because the testers do not have enough knowledge. To evaluate this, the respondents were asked to rank the test department’s knowledge on a scale from one to five (with five being high) in test theory, system design, and how the system will be used (domain knowledge). Figure 8 shows the responses from each organization for the three types of knowledge. Organizations that produce embedded software have shaded bars.

3.5 Test Time Consumption

Figure 9 shows the amount of person-time spent on testing, relative to the total development time. Organizations 7, 9, and 10 are the only ones who actually measured this (highlighted with white bars in the figure), the others are estimates. The representative from the first organization had no record of the amount of testing and was unwilling to make an estimation. As said previously, organization 10 has two types of projects, one type with very little new functionality (10a) and one with mostly new functionality (10b).

These observations generally agree with old observations that testing consumes a major part of the resources in development projects [4, 7, 10, 20]. Two organizations (7 and 10), both of which produce safety-critical embedded software, report measured test time consumptions of 65%.

There was a wide variation in time used on testing. Three were fairly low (less than 15%), two were high, and the rest were between 30% and 45%, which is close to the mean (35.1%).

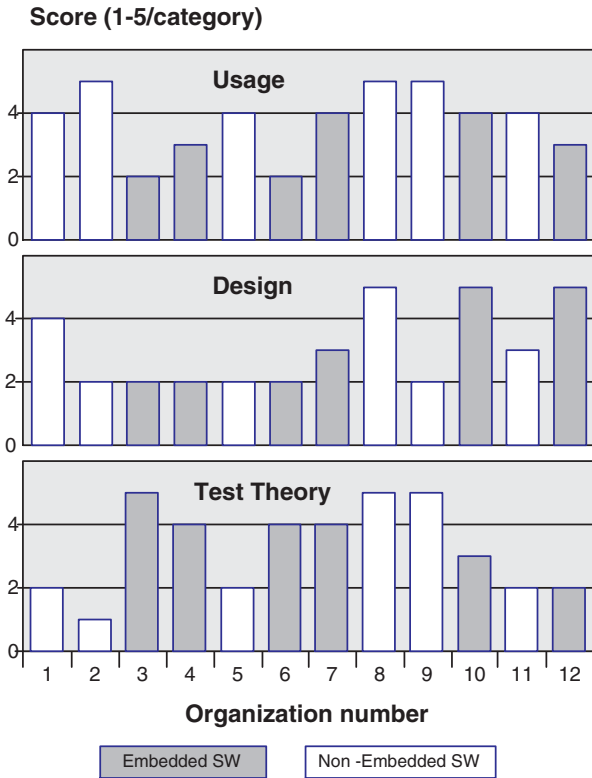


Figure 8. Three types of test team knowledge.

Given the total amount of time spent on testing, the next question was to investigate how this time was spent across the different test phases. This study defined the test phases to be *pre-execution*, *component* (including unit testing by the programmers), *integration*, *system*, *acceptance*, and *other*. The last category was used by two organizations for *field testing*, where the product is tested in the deployment environment. All respondents successfully mapped their test process onto these phases. Nine organizations have separate test teams to perform system testing, two of whom also perform integration.

Figure 10 shows how time for testing activities was distributed across the different test phases. This is a “box-plot” graph. Each box represents the 50% of the values in the middle and the lines above and below the boxes extend to the highest and lowest value. For example, in the pre-execution phase the organization that reported the least amount of time spent was 5%, and the highest was 40%. The box ranges from 10% to 35%, so one quarter of the values were below 10% and one quarter above 35%. The diamonds represent the mean value reported (19% for pre-execution).

Of the twelve organizations, two had neither knowledge nor estimates. One of the remaining ten had substantiated

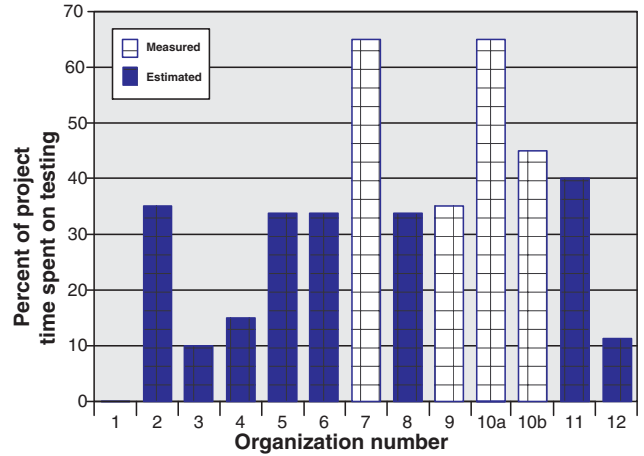


Figure 9. Time spent on testing, relative to other development activities.

information, but did not want to share this information, so the figure includes data for only nine organizations. Six of the nine organizations spend more testing time in the system testing phase than in any other testing phase. Seven of the nine organizations spend 50% or more of their total testing time in one single phase. Overall, it is striking how little time is spent on test activities early in development.

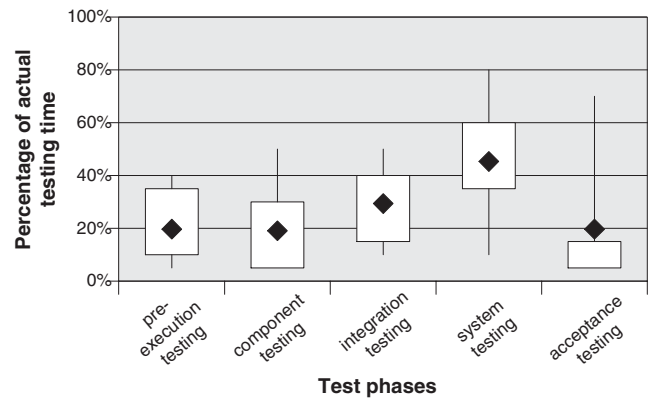


Figure 10. Distribution of testing time over test phases.

3.6 Software Development Metrics

Metrics are used to substantiate claims about the tested product as well as the status of the project. Metrics also help managers decide if a process change has helped.

Two organizations do not collect any metrics at all. Of the remaining ten, all monitor resources used, usually time.

Nine monitor test progress, usually test cases executed, and nine (not the same nine) monitor defects found. These metrics are collected and used within the projects.

Although a few organizations monitor some aspect of project performance, e.g. requirements coverage, only one organization has an established metrics program that allows them to evaluate changes in processes, tools, methods, etc.

Within the system lifetime, the most common metric is defects found during operation, which is monitored by six organizations. Two of these also measure effectiveness and how much time is spend in maintenance. Four organizations use their metrics to compare projects or products.

4 Analysis and Results

This section analyzes the data presented in Section 3, following the order of the research questions in section 2.1. Wherever applicable our results are compared with and contrasted to two other recent testing state-of-practice investigations. One was conducted in Australia during 2002 and 2003 [17] and the other was conducted in Alberta, Canada during 2002 [12].

4.1 Test Case Selection Methods

The three organizations that use test strategies to help select test cases use basic methods (equivalence partitioning, boundary value analysis and requirements coverage). None reported using even simple test criteria like edge coverage on graphs, let alone more advanced criteria such as MCDC, data flow or mutation.

These findings correspond with results from both the Australian and Canadian studies. In the Australian investigation [17], 29 of 64 organizations reported using a black-box method during the past three years. Only 16 reported using white-box methods and 3 reported using mutation analysis.

There are differences in methodology between our study and the Australian study that may inflate some of the Australian numbers relative to ours. The 64 participants in the Australian survey responded to a massively distributed inquiry that was sent to well over 10,000 Australian IT professionals. Exactly half of the respondents came from software houses and IT consultancy businesses. It seems reasonable to expect that relatively immature organizations are **less** inclined to participate in such a study, and it is also likely that test maturity is higher in consultancy companies than in many other industries. Another difference is that our survey asked which methods are being used at the present time, while in the Australian investigation they asked which methods had been used the past three years.

In the Canadian study [12], the most frequently used structured test case selection method is boundary value

analysis used by slightly less than 30% of the respondents.

The Canadian study, like the Australia study, was based on a mail survey form. Slightly less than 60 respondents participated in the survey. Survey candidates were chosen randomly. The paper did not report how many requests were sent out so it is impossible to judge the representativity of the sample. Most respondents represented organizations that develop business information systems.

One conclusion that can clearly be drawn from the three studies is that the use of structured test case of methods is very limited. Finding similar patterns in three independent investigations in three different parts of the world strengthens the belief that this situation applies in other parts of the world as well.

Many managers expressed concerns for the lack of structure in the testing. This makes us believe that managers recognize the benefits of structured testing. However, this is far from a commitment to improve.

If managers want to improve their testing, they face at least three major obstacles. First, the organizations are all commercially successful, indicating that their products are at least "good enough." The upper management may therefore be reluctant to invest in change unless the test managers can present a case based on hard facts. The second obstacle is that most organizations in this study do not record enough metrics to describe the current situation in economic terms. Hence the payoff from improved testing cannot be quantified.

A third obstacle preventing change is that most types of improvements require an initial investment that would (hopefully) pay back later. With tight project schedules and short times-to-market, it is hard to convince program managers to select a project to try a new process on.

The Australian investigation explored the respondents' perceptions of possible barriers to adoption of structured testing methods. Lack of expertise was ranked highest (28 votes) followed by time-consumption (20 votes) and lack of support tools (18 votes). (The respondents were allowed to choose more than one.) The authors' main conclusion from these data is that either software professionals are not given proper education in universities or in industry, or there is a genuine shortage of software testing professionals. In either case, there is an obvious need for more software testing education.

In our investigation, the organizations ranked test theory knowledge as being on average *fair*, which could mean that lack of test knowledge is a significant contributor to the lack of test maturity.

4.2 Test Strategy

Section 3.2 showed that nine of the twelve organizations have some notion of test strategy. This may seem positive,

but the contents of these test strategies and how they are used is less positive. Only three organizations maintain information about how testing should be performed. Thus, only three have information in their test strategies that is normally considered to be test strategy information. The other types of information maintained in the test strategies of the organizations are certainly important but normally maintained in other documents, for instance test and trouble reporting processes.

Insufficient use of test strategies does not always lead to poor products. Test strategy decisions are continuously made during the project and if the organization is lucky or if the decisions makers are good, the decisions can still result in good products. Also, hard work (that is, extra time and money) can often make up for poor strategies. Without a test strategy to guide the decision making process, there is a risk that product and process quality varies greatly between different projects. Organizations also become more dependent on key persons to achieve the project goals. Suggestion for, implementation of, and in particular, evaluation of improvement also become more difficult.

Of the three organizations that have test strategies, we only have resource consumption data for two. An interesting similarity of the two is that they report almost identical distribution of test time in figure 10. Even more interesting is that both organizations invest 35% to 40% of the total test time in pre-execution testing, which is twice as much as any other organization in the study. Obviously, the number of observations is far too low to make any conclusions but it is not surprising that test strategies may result in more test activities early in development.

4.3 Moment of Involvement

One result that surprised us is that most organizations involve their testers early in the projects; half from the project start. Another four organizations involve their testers during the requirements collection. Only two of twelve brought in testers at the end, which happily contradicts the “throwing software over the wall” process that is sometimes assumed.

In the Canadian investigation [12], almost 70% of the participants were involved in requirements management activities. More than 60% were involved in software quality assurance. The interpretation of this is that people take on multiple roles and that combining testing and requirements work is not uncommon.

The claim of early involvement is validated by the observation that on average, 14% of the total execution time is spent on pre-execution testing for the nine organizations where this information is available, as shown in figure 9. Thus, the suspicion that the lack of test maturity stems from late involvement of the testers does not seem to be correct.

The two organizations that do not involve their testers

until the end of implementation have several properties in common. Their development organizations are very small, 15 – 25 people. Their projects are short, four to six months and 1000 to 3000 person-hours. They also estimate that their testers have little knowledge in test theory. Instead, many of their testers have a background as users of the systems, which is reflected in high domain knowledge of the test teams. Finally, these are the only two organizations that do not use test specifications nor produce final test reports. Our suspicion is that high domain knowledge may compensate, up to a certain level, for lack of testing theory. Also, a low level of test maturity may be safer with small projects than with larger projects.

4.4 Test Team Knowledge

It is interesting to note that the evaluation of test team knowledge is very different in organizations that develop embedded software from the organizations that develop non-embedded software. Embedded software organizations rank the test theory knowledge as much higher and the non-embedded software organizations rank the domain knowledge as higher. The average test theory knowledge for embedded organizations is 3.67, while for non-embedded organizations it is only 2.83. The average domain knowledge for embedded organizations is only 3.0, while for non-embedded organizations it is 4.5.

Our interpretation of these data is that embedded and non-embedded systems may be tested in different ways. There also seem to exist at least two different approaches to testing; one based on using a high level of domain knowledge and the other using a highly refined method for generating tests.

The results reported in the Canadian investigation can be interpreted to point in the same direction. Most of the respondents come from organizations that develop business software. They report that only 25% of the respondents have received some sort of testing training at their work. This type of knowledge would typically fall into the area of test theory. Further, they report that “tester’s skill and intuition” is the primary method for test case selection. This is likely to include at least some domain knowledge.

4.5 Test Time Consumption

Our findings match earlier studies, which found that testing consumes a large amount of resources [4, 7, 10, 20]. Further, we believe that the average (35%) amount of time spent on testing shown in figure 9 is low due to the fact that only three organizations (four values) are based on actual data, and all these values are above the average, with three of them being the highest values. When guessing, it is easy

to overlook less obvious contributions to the values, so others may be higher.

Both organizations that spend 65% of their times on testing are large and old organizations that produce both hardware and software for safety critical systems. It is not surprising that companies spend more time testing safety critical software.

A test strategy seems to help the organizations spend time more evenly over the different test phases. This is illustrated by the two organizations that have explicit test strategies being the only two that have an even distribution of test time over the different phases. The other seven organizations spend most of their test time in system testing as shown in figure 10, with a glaring fact that the average system testing time is 46%. This is probably indicative of a lack of test maturity, and suggests there is a lot of room for improvement if the testing effort could be more evenly distributed.

It is well known that system testing is the most expensive time to find failures. Moreover, it is much harder to debug failures (tracing back to actual software faults) when the failures are found at the system level than when found during component testing.

The study of one of the two organizations that spends 65% of their time on testing led to an important insight. Projects that added a small amount of new functionality emphasized a lot of regression testing, and the relative amount of testing was **higher** than with projects that had mostly new functionality.

A Spearman test shows a strong correlation (0.78) between the amount of documentation produced and the time spent on test preparation. This is hardly surprising since most documents (test specifications) are produced during this phase. There is a medium strength correlation (0.55) between the amount of information in the test specifications and the time spent on test preparation.

There is a medium strong inverse correlation (-0.53) between system testing time and time spent on test preparation and also a medium strong correlation (0.54) between system testing time and test execution. Our conclusion from these findings is that good preparations and possibly re-use of previously generated documentation may help cut overall testing time.

In the Canadian investigation [12], most respondents reported that they spent between 10% and 30% of their work on pre-release testing. Their definition of testing does not include activities prior to component testing.

4.6 Metrics

In section 3.6, it is shown that except for one organization, the collected metrics only allow limited evaluations of the product within the project. This result also corresponds

to the findings in the Australian investigation [17] where the most common metric was number of defects found used by 31 of the 65 surveyed respondents.

An obvious conclusion from these investigations is that there is plenty of room for improvement in the area of metrics. However, it is still open as to which metrics are best and how to capture them.

5 Summary and Conclusions

This paper presents data from a detailed study of how twelve organizations test software. Following is a summary list of our major findings.

1. An average of 35% of development time is spent on testing (with a significant amount of variation)
2. Structured testing strategies and test selection methods are not widely used
3. The majority of tests are run at the system level; relatively little unit and integration testing is done
4. Projects with little new development spend more development time on testing than projects with mostly new development
5. The use of an explicit test strategy seems to help organizations improve in general
6. Test preparations and re-use may help cut overall costs for system testing

The overall test maturity of the organizations was found to be low; only three use structured test strategies in the proper sense and only three use structured test case selection methods. However, many of the organizations are commercially successful, leading us to conclude that their products are still “good enough” in an economic sense. This could be because the individual testers are very good, market expectations are low, the organizations overcome poor process with hard work (which implies that their testing is inefficient), or because structured testing is not helpful. An open question is to what extent can testers with good domain knowledge compensate for lack of structure in testing?

From a high level management point of view, as long as the products make a profit, there is little motivation for investing in improved testing. Many of the managers we interviewed expressed concerns for the relatively low test maturity, which means that there is an awareness of a problem. Another reason that change is slow is because it is hard to assess the current situation and estimate the potential gain. Motivation for improved quality is often driven by competition (as in web applications) and by standard organizations.

We find it surprising that there are not stricter standards for testing of safety critical software.

The organizations in this study that were found to be the most mature all use more detailed test strategies that describe test phases and which test case selection methods to use detailed test strategies. Thus, we conclude that the use of a structured testing strategy is a great help for organizations to increase their testing maturity.

Based on the findings in this study we advise test managers to concentrate on implementing a metrics program that allows for project and product quality assessments in economic terms.

One of our most important observation flows from summary finding number 4, especially when combined with one of the most important trends in software design and development today. When companies spend less time developing new software, they spend more of their budget on testing. This is especially significant because software development organizations are dramatically increasing the use of component integration and reuse by “wiring together” pre-existing components, meaning less new software is being written all the time. Although somewhat speculative, these observations suggest that we are on the verge of an economic “phase transition,” and significant investments in testing can now have a **major impact** on companies’ economic success. In fact, testing is becoming the prime economic driver in software process. Indirect evidence for this can already be seen from a recent increase in the number of industry-oriented books on software testing, widespread adoption of tools for component (unit) testing such as *JUnit*, and the recent success of testing tools such as *Agitar’s Agitator*.

6 Acknowledgments

We would like to thank the participating organizations for their openness and Enea for providing the contacts. Then we would like to thank Åsa G. Dahlstedt, Anders Claesson, Aynur Abdurazik, Wuzhi Xu and the anonymous reviewers for all the help. The first author is sponsored in part by Enea AB and the Swedish Knowledge Foundation. The second author is sponsored in part by National Institute of Standards and Technology (NIST), Software Diagnostics and Conformance Testing Division (SDCT).

References

- [1] D. G. Altman. *Practical Statistics for Medical Research*. Chapman & Hall (CRC), London, 1991.
- [2] P. E. Ammann and J. Offutt. Using formal methods to derive test frames in category-partition testing. In *Proceedings of the Ninth Annual Conference on Computer Assurance (COMPASS’94)*, Gaithersburg MD, pages 69–80. IEEE Computer Society Press, June 1994.
- [3] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, 1990.
- [4] B. Boehm. Some Information Processing Implications of Air Force Space Missions: 1970-1980, Santa Monica, California, The Rand Corp. Memorandum RM-6213-PR.
- [5] L. Bratthall, P. Runesson, K. Adelswård, and W. Eriksson. A survey of lead-time challenges in the development and evolution of distributed real-time systems. *Information and Software Technology*, 42:947–958, 2000.
- [6] British Computer Society BS 7925-1. *Glossary of terms used in software testing*, 1998.
- [7] F. Brooks. *The Mythical Man-Month*. Addison-Wesley: Reading MA, 1975.
- [8] T. Cook and D. Campbell. *Quasi-Experimentation Design and Analysis Issues for Field Settings*. Houghton Mifflin Company, 1979.
- [9] R. Craig and S. Jaskiel. *Systematic Software Testing*. Artech House Publishers, 2002.
- [10] M. Deutsch. Verification and validation. In Jensen and Tonies, editors, *Software Engineering*. Prentice Hall, 1987.
- [11] E. Dustin. *Effective Software Testing: 50 Specific Ways to Improve Your Testing*. Addison-Wesley, 2002.
- [12] A. Geras, M. Smith, and J. Miller. A survey of software testing practices in Alberta. *Canadian Journal of Electrical and Computer Engineering*, 29(3):183–191, 2004.
- [13] M. Grindal, A. J. Offutt, and S. F. Andler. Combination testing strategies: A survey. *Software Testing, Verification, and Reliability*, 15(3):167–199, September 2005.
- [14] M. Grindal, J. Offutt, and J. Mellin. On the testing maturity of software producing organizations: Detailed data. Technical report ISE-TR-06-03, Department of Information and Software Engineering, George Mason University, Fairfax, VA, July 2006. <http://www.ise.gmu.edu/techrep/>.
- [15] T. Koomen and M. Pol. *Test Process Improvement - A practical step-by-step guide to structured testing*. ACM Press, 1999.
- [16] G. J. Myers. *The Art of Software Testing*. John Wiley and Sons, 1979.
- [17] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen. A preliminary survey on software testing practices in Australia. In *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC04)*, April 13-16, Melbourne, Australia, pages 116–125. IEEE, April 2004.
- [18] J. Offutt. Quality attributes of Web software applications. *IEEE Software: Special Issue on Software Engineering of Internet Software*, 19(2):25–32, March/April 2002.
- [19] P. Runeson, C. Andersson, and M. Höst. Test Processes in software product evolution - a qualitative survey on the state of practice. *Journal of Software Maintenance and Evolution: Research and Practice*, 15(1):41–59, jan-feb 2003.
- [20] E. Yourdon. *Techniques of Program Structure and Design*. Prentice Hall Inc., 1975.