On the Testing Maturity of Software Producing Organizations: Detailed Data

Mats Grindal
Enea AB, Box 1033, SE-164 21 Kista, Sweden
and
School of Humanities and Informatics
University of Skövde, Sweden
magr@enea.se

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

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

April 26, 2006

Technical Report ISE-TR-06-03
Department of Information and Software Engineering, George Mason University

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.

Contents

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% [Boe, Bro75, Deu87, You75]. 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 [BRAE00].

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 [Off02].

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. In particular, 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 [Mye79, Bei90], but are rarely used in industry. This study also reasons about the factors that influence the application of testing research results in industry.

An early decision of this study was to focus on a diverse set of organizations instead of one type of organization. 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 the 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 was collected, and how the analysis of the data was conducted. This section also discusses aspects of validity with respect to this study. Section 3 presents our collected data and section 4 analyzes this 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 different organizations. It can be viewed as a qualitative study with some quantitative parts. The forthcoming sections describe in more detail how this study was carried out.

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.

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 one 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 a lot of 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 products, 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,

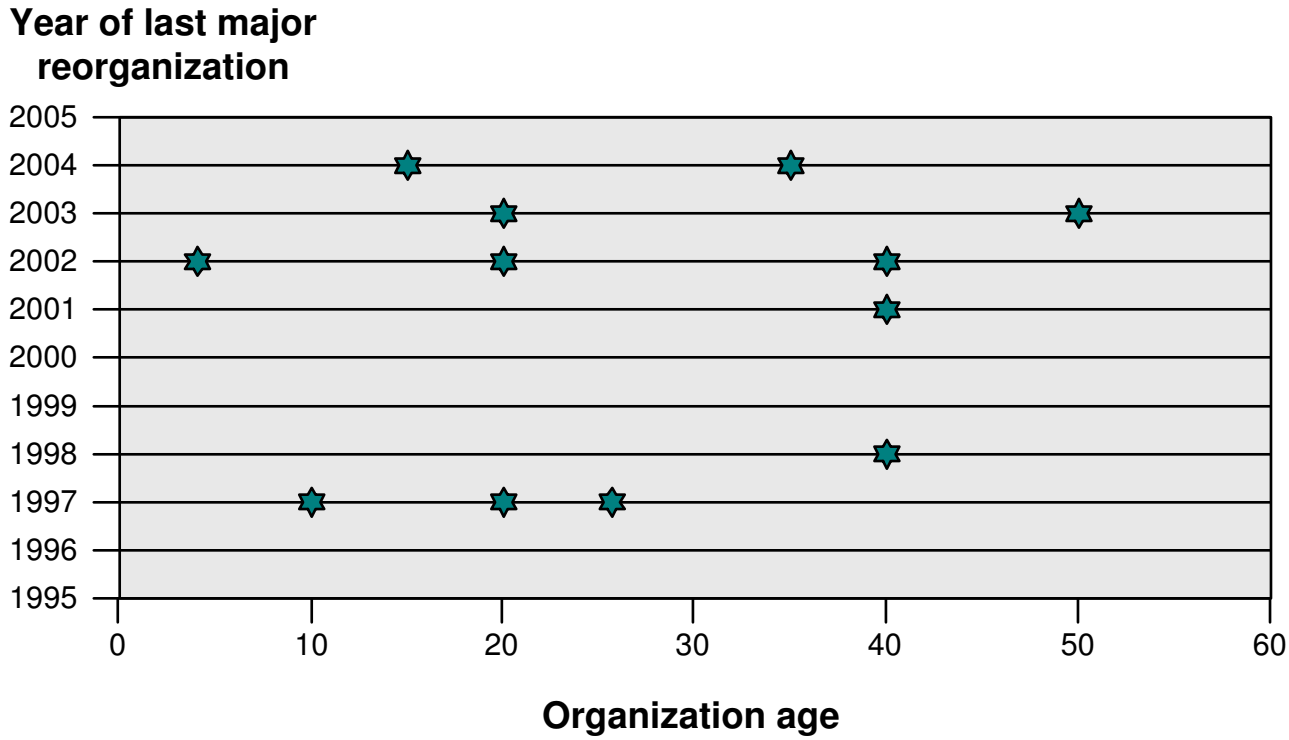

Figure 1: Age and year of last reorganization.

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.

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.

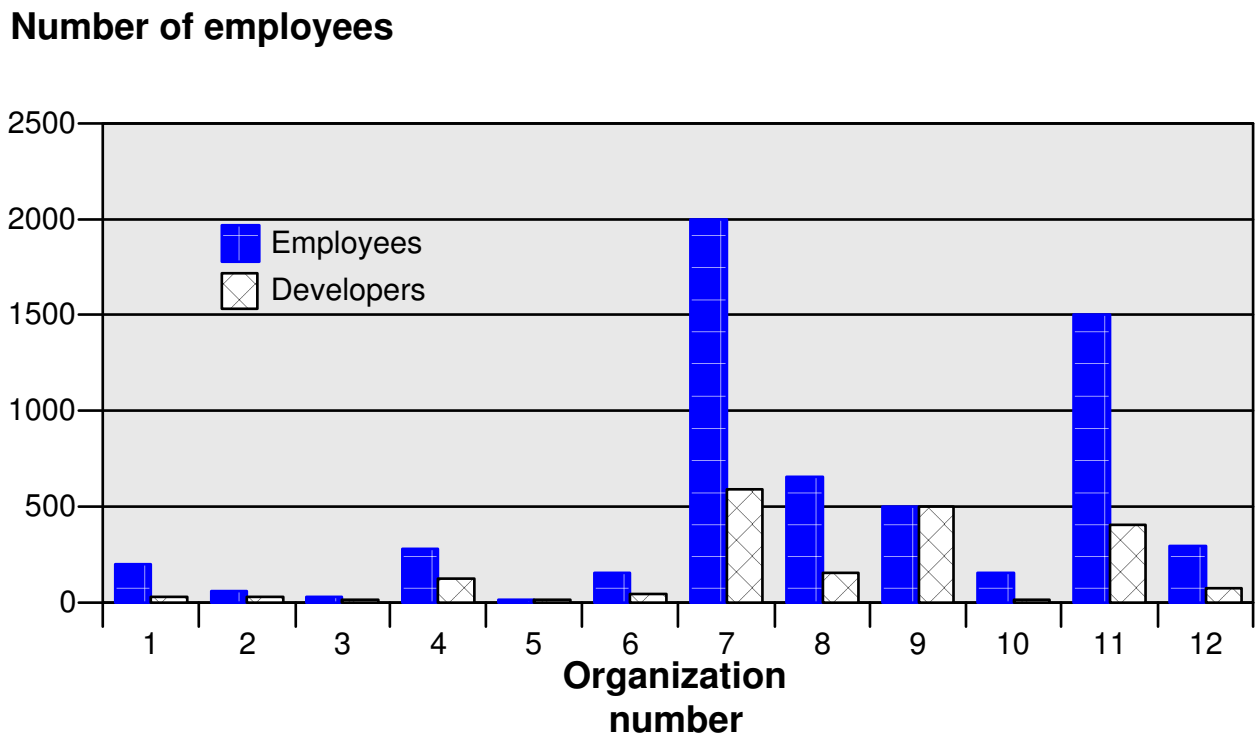

Figure 2: Total number of employees and number who are developers.

2.4 Analysis

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

The graphs were then used to identify differences and similarities among the organizations. Cross-property comparisons were then performed through Spearman tests [Alt91]. 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.

2.5 Validity

Cook and Campbell [CC79] 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.

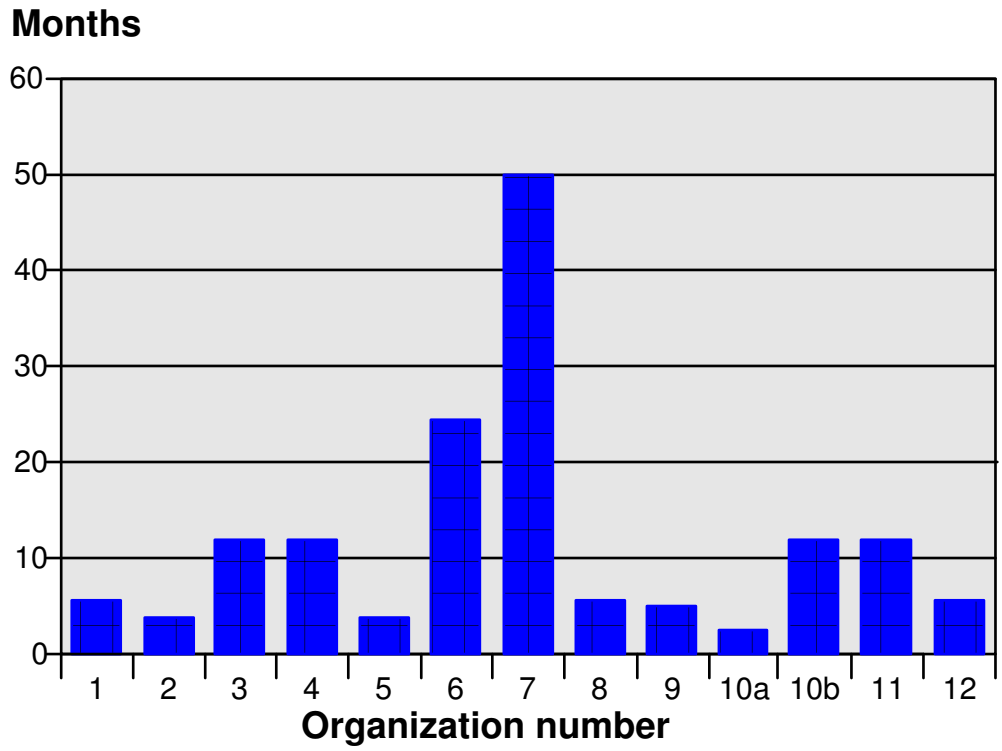

Figure 3: Size of projects in terms of calendar time.

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 documented and communicated via the management.

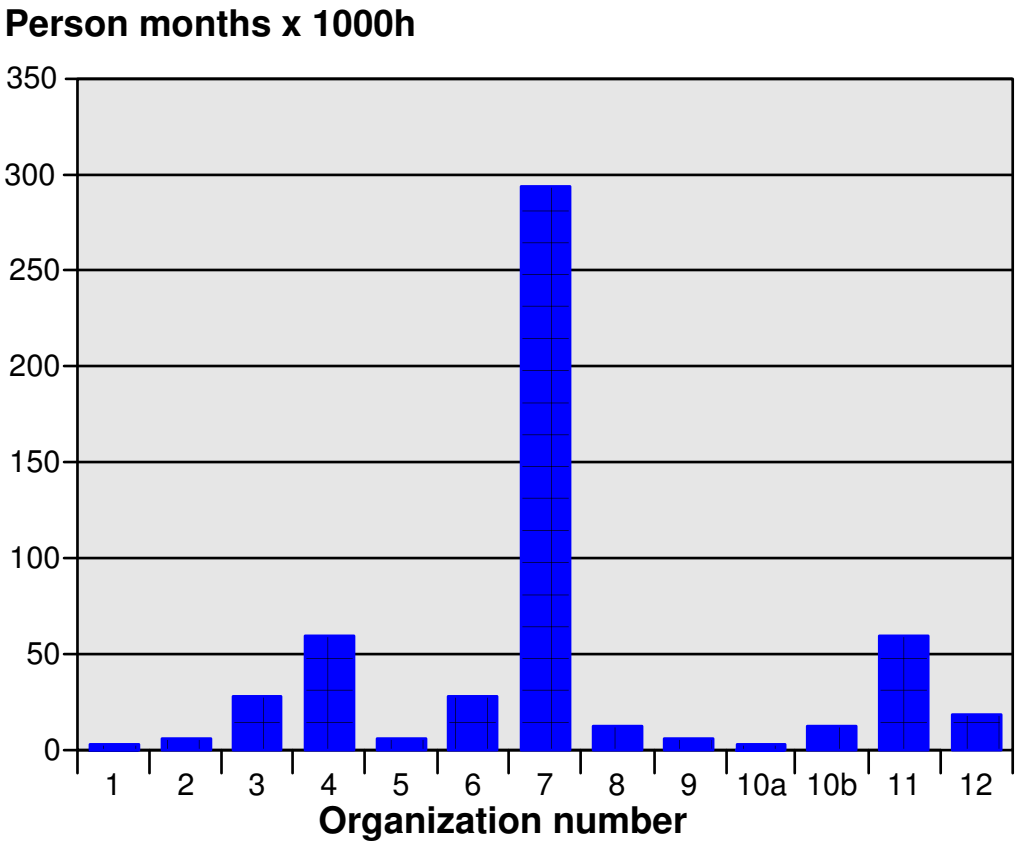

Figure 4: Size of projects in terms of person time.

Thus, what management thinks is being use 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) [KP99] and BS7925-1 [BS 98]. 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 not the whole picture is revealed. This was partly addressed by having overlapping questions to be able to detect possible inconsistencies in the answers.

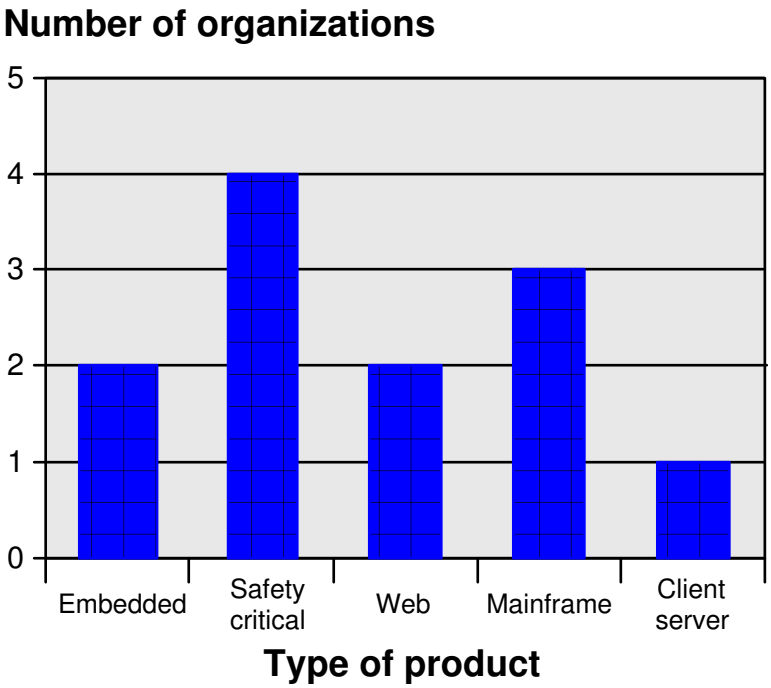

Figure 5: Types of products developed by each organization.

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. External validity of studies like this one is inherently difficult to judge 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.

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: Interpretation of Spearman coefficients.

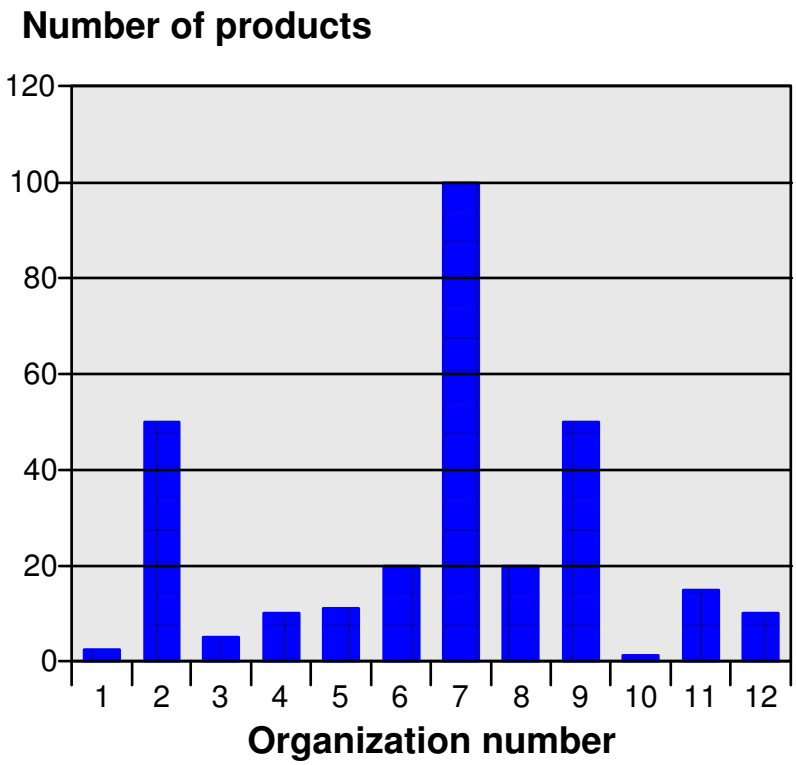

Figure 6: The number of products developed by each organization.

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 the way 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 in other parts of the world.

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.

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 [Mye79], boundary value analysis [Mye79] and some basic combination strategies [GOA05], for instance "each choice" [AO94]. Organization 8 uses boundary value analysis and cause-effect graphing [Mye79] 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 [KP99]. 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.

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 [Dus02, CJ02]. 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.

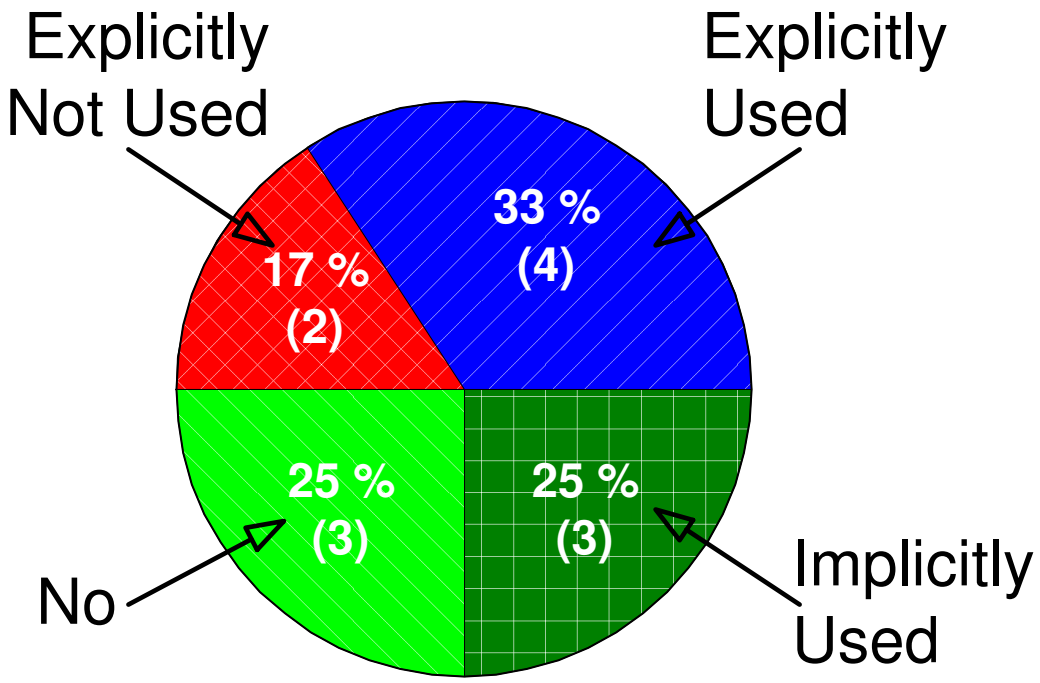

Figure 7: Does a testing strategy exist (explicitly or implicitly)?

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.

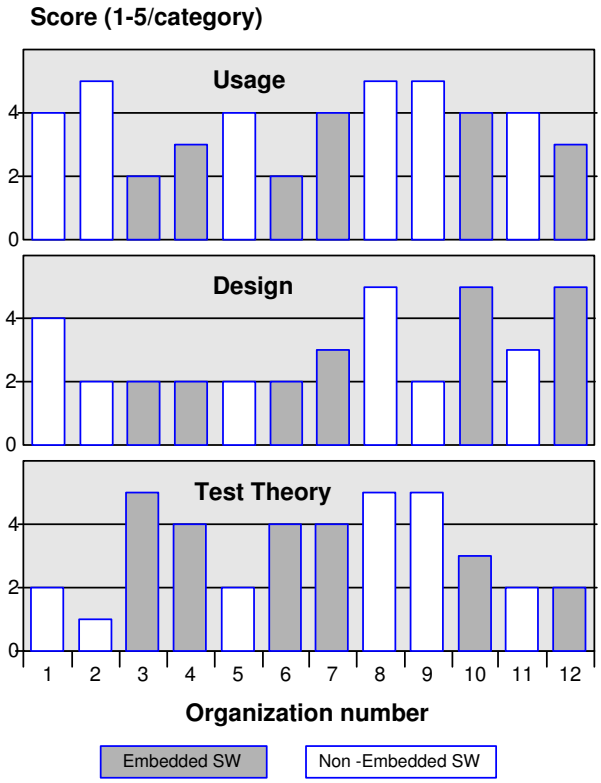

Figure 8: Three types of test team knowledge. Shaded bars represent organizations producing embedded software and non-shaded bars non-embedded software

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 a lot of new functionality (10b).

These observations generally agree with old observations that testing consumes a major part of the resources in development projects [Boe, Bro75, Deu87, You75]. 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

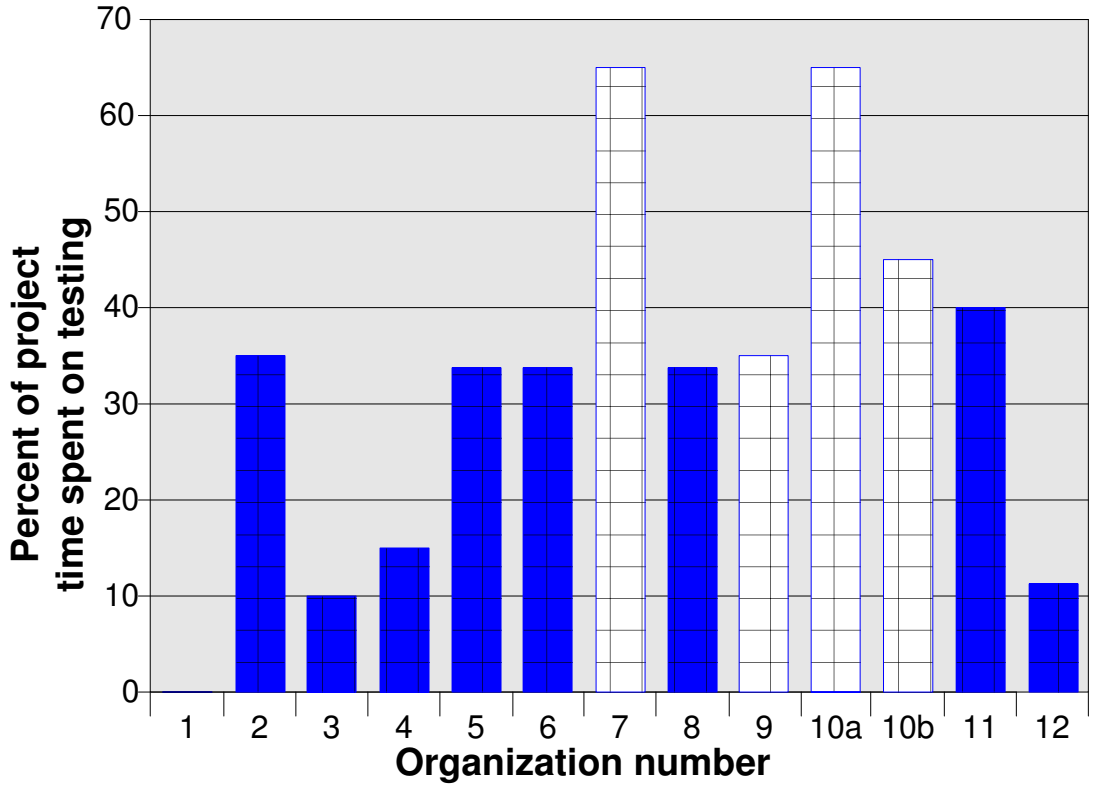

Figure 9: Time spent on testing, relative to other development activities. White bars represent organizations who explicitly measured the time.

were high, and the rest were between 30% and 45%, which is close to the mean (35.1%).

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 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 in early life-cycle test activities.

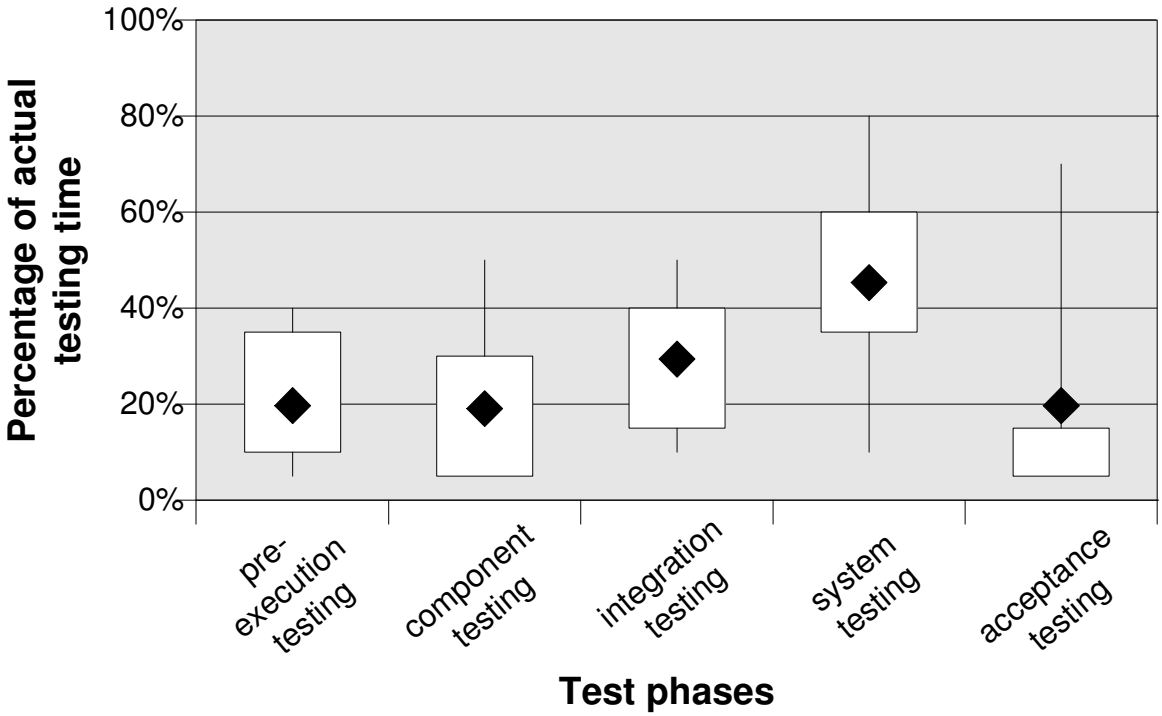

Figure 10: Distribution of testing time over different test phases. Lines show the high and low values, boxes show the middle 50% values, and diamonds show the mean values.

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 helps managers decide if a process change has helped.

Two of the organizations do not collect any metrics at all. Between the remaining ten, all monitor used resources, usually time. Nine monitor test progress, usually executed test cases, and

nine, but not the same nine, monitor found defects. All 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 the results of a testing state-of-practice investigation of Australian software producing companies that was conducted during 2002 and 2003 [NMR+04].

4.1 Test Case Selection Methods

The three organizations that use test strategies to help select test cases use very basic methods (equivalence partitioning, boundary value analysis and requirements coverage). None of the organizations 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 a recent study in Australia [NMR+04]. In the Australian investigation, 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. 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 that 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.

One conclusion that can clearly be drawn from both studies is that the use of structured test case methods both among the organizations in both studies survey is very limited. Finding similar patterns in two independent investigations in two different parts of the world strengthen the

suspicion 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 superficially, 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 data on resource consumption 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 are far too low to make any conclusions but it is not surprising that test strategies may help to distribute the testing time more evenly.

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 to occur.

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 this 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.

4.5 Test Time Consumption

Our findings match earlier studies, which found that testing consumes a large amount of resources [Boe, Bro75, Deu87, You75]. 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 solid facts, 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 system. 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 more 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 in 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.

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 [NMR+04] where the most common metric was number of found defects 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 they are best captured.

5 Summary and Conclusions

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

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 a higher percentage of their development time on testing than projects with lots of new development
5. The use of an explicit test strategy seem 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 knowledge in the how the product is used compensate for lack of structure in testing?

From a high level management point of view, as long as the products make a profit, there are not a lot of compelling reasons to invest 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.

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 point number 4 above, 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, meaning less new software is being written all the time. Instead, we are spending our development efforts in "wiring together" pre-existing components. 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 book on software testing, widespread adoption of tools for developer (unit) testing such as *JUnit*, and the recent success of testing tools such as *Agitar's Agitator*.

6 Acknowledgments

We would like to thank Enea for generously providing the contacts to the organizations investigated. Then we owe each of the participating organizations our gratitude. Thank you for your openness and generosity! Then we would like to thank Åsa G. Dahlstedt for reviewing the questionnaire. The authors also would like to thank Anders Claesson, Aynur Abdurazik, and Wuzhi Xu for help with references and review. The first author is sponsored in part by Enea AB and 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

[Alt91] D. G. Altman. *Practical Statistics for Medical Research.* Chapman & Hall (CRC), London, 1991.

[AO94] Paul E. Ammann and Jeff 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.

[Bei90] Boris Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, 1990.

[Boe] B.W. Boehm. Some Information Processing Implications of Air Force Space Missions: 1970-1980, Santa Monica, California, The Rand Corp. Memorandum RM-6213-PR.

[BRAE00] L. Bratthall, P. Runesson, K. Adelswärd, and W. Eriksson. A survey of lead-time challenges ini the development and evolution of distributed real-time systems. *Information and Software Technology*, 42:947–958, 2000.

[Bro75] F.P. Brooks. *The Mythical Man-Month*. Addison-Wesley: Reading MA, 1975.

[BS 98] British Computer Society BS 7925-1. *Glossary of terms used in software testing*, 1998.

[CC79] T.D. Cook and D.T. Campbell. *Quasi-Experimentation Design and Analysis Issues for Field Settings*. Houghton Mifflin Company, 1979.

[CJ02] R.D. Craig and S.P. Jaskiel. *Systematic Software Testing*. Artech House Publishers, 2002.

[Deu87] M.S. Deutsch. Verification and validation. In Jensen and Tonies, editors, *Software Engineering*. Prentice Hall, 1987.

[Dus02] E. Dustin. *Effective Software Testing: 50 Specific Ways to Improve Your Testing*. Addison-Wesley, 2002.

[GOA05] 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.

[KP99] Tim Koomen and Martin Pol. *Test Process Improvement - A practical step-by-step guide to structured testing*. ACM Press, 1999.

[Mye79] Glenford J. Myers. *The Art of Software Testing*. John Wiley and Sons, 1979.

[NMR+04] S.P. Ng, T. Murnane, K. Reed, D. Grant, and T.Y. Chen. A preliminary survey on software testing practices in Aaustralia. In *Proceedings of the 2004 Australian Software Engineering Conference (ASWEC04), April 13-16, Melbourne, Australia*, pages 116–125. IEEE, April 2004.

[Off02] Jeff Offutt. Quality attributes of Web software applications. *IEEE Software: Special Issue on Software Engineering of Internet Software*, 19(2):25–32, March/April 2002.

[You75] E. Yourdon. *Techniques of Program Structure and Design.* Prentice Hall Inc., 1975.

A Appendix A - The Questionnaire

Appendix A lists the questions that were used in the questionnaire.

Questionnaire

1. Age of organization
When was the organization founded?
When was the latest re-organization?

2. Size of organization
How many employees are there in the organization?
How many employees are involved in product development?
What is the geographic organization of the organization?

3. Product
What type of product is being developed?
How many products are being developed?

4. Test Activities in the development process
Which test activities occurs in a project?

5. Test organization
Does a separate test organization exist?
What is the competence of the test organization:
Test Theory(1-5): System Design(1-5): System Usage(1-5):
When is the test organization involved in the project:
start of project/requirements collection/start of implementation/end of implementation

6. Project
What is the calendar time duration of projects?
What is the man-hour duration of projects?
How much of the project cost is cost for testing?

7. Testware
Which of the following documents are produced by the (system) test organization:
test plan/test specification/test status report/trouble report/test delivery note/final test report

8. Test strategy
Does a testing strategy exist (explicitly/implicitly)?
Is the testing strategy used in the projects?
What is included in the testing strategy?

9. Test methods
Are testing methods used in the projects?
Which testing methods are used?
How are testing methods selected?

10. Test Cases
How are test cases selected?
Are test cases reused from previous projects?
On what grounds?
How are test cases stored?
Which of the following information are included in test cases:
Sensible Name/ Unique Id/ Revision/ Priority/ Author/ Goal/ Purpose/ Configuration Requirements/ Precondition/ Action(s)/ Result(s)/ Postcondition/ Type of Test Case/ TC Passing Criteria

11. Metrics
Are any metrics collected?
Which metrics?
(project/product):
- used resources (hours, money, etc.)
- performed activities (hours, lead time, etc.)
- size and complexity of tested system
- test products (test specs, test cases, etc.)
- test progress (performed tests, status)
- number of defects (sorted by different properties)

(project/process):
- defect find-effectiveness
- defect find-efficiency
- test coverage
- testware defects
- perception of quality

(product in operation):
- found faults in production

effectiveness of maintenance
efficiency of maintenance
(cross product comparison):
collection and comparison of metrics across different products

12. Cost
What is the total cost of testing in the project?
How much of the total cost of testing is spent on
pre-execution testing
component testing
integration testing
system testing
acceptance testing
How much of the system testing cost is spent on
planning
preparation
execution

B Question 1 - Age

Question 1 concerns different aspects of organizational age (Table 2).

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Org. Age	25+	10	14	35	4	20	40	20	40	50	20	40
Latest Reorg.	1997	1997	2004	2004	2002	2002	2001	1997	2002	2003	2003	1998

Table 2: Age of each company and year of the latest reorganization

C Question 2 - Size

Question 2 concerns different aspects of organizational size. (Tables 3 and 4).

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Total	200	60	35	300	15	150	2000	650	500	160	1500	300
Development	25	35	30	120	15	40	600	150	500	20	400	70

Table 3: Number of employees in total and in the development organization

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Dev. sites	2	1	1	1	1	1	4	2	2	5	5	1

Table 4: Number of development sites

D Question 3 - Type of Product

Question 3 concerns the types of products that are being developed. (Table 5). In several cases an organization develops more than one type of product. In such cases, the interview focused on the main product of that organization.

	organization					
	1	**2**	**3**	**4**	**5**	**6**
Product type	SW Mf.	SW CS	HW/SW Emb.	HW/SW SC	SW Mf.	HW/SW SC
	organization					
	7	**8**	**9**	**10**	**11**	**12**
Product type	HW/SW SC	SW Web	SW Mf.	HW/SW SC	SW Web	HW/SW Emb.

Table 5: Type of product: Mf-Mainframe, CS-Client Server, Emb-Embedded, SC-Safety Critical

Question 3 also covers the number of products or versions of products each organization develops concurrently. (Table 6).

E Question 4 - Development Process

Question 4 concerns the test phases each organization maintains in their standard projects. (Table 7).

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Nrof Prods.	3	50	5	9	10	20	100	20	50	2	15	9

Table 6: Number of products or versions of products concurrently developed

	organization											
Test Phase	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Pre-execution			X	X		X	X	X	X	X	X	
Unit Testing	X	X	X	X		X	X	X	X	X	X	X
Integration Testing			X	X	X			X				X
System Testing	X	X	X	X	X	X	X	X	X	X	X	X
Acceptance Testing	X			X				X	X			X
Field Testing			X								X	

Table 7: Formalized test activities throughout the development process

F Question 5 - Test Organization

Question 5 is focused on when testers are actively involved in the project (table 8), whether or not the organizations have a separate testing organization (table 9), and the knowledge of the people doing system testing (table 10).

moment	**organizations**
start of project	4,7,9,10,11,12
requirements collection	2,3,6,8
start of implementation	
end of implementation	1,5

Table 8: When are testers actively involved in development projects? Numbers refer to organizations.

G Question 6 - Project Duration

Question 6 concerns project duration in terms of calender and person time (table 11). Relative cost of testing, i.e., the amount of resources in a project that are dedicated to test activities (table 12).

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Separate Test Org.	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No

Table 9: Formalized test activities throughout the development process

	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Test Theory	2	1	5	4	2	4	4	5	5	3	2	2
Prod. Design	4	2	2	2	2	2	3	5	2	5	3	5
End-usage	4	5	2	3	4	2	4	5	5	4	4	3

Table 10: Estimated knowledge (1(low) - 5(high)) of system testers in three knowledge areas

H Question 7 - Testware

Question 7 concerns which types of test documentation are written in the projects. (Table 13)

I Question 8 - Test Strategy

Question 8 covers the test strategy. Does a test strategy exist (implicitly or explicitly) and is it used? (Table 14) If there is a test strategy, what does it contain? (Table 15)

J Question 9 - Test Methods

Question 9 is focused on test methods. Which test methods are being used and how does a tester know which test method to apply? (Table 16)

	organization												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10a**	**10b**	**11**	**12**
Calender months	6	4	12	12	4	24	50	6	4,5	3	12	12	6
Person hours (x1000)	1	4	36	60	3	30	288	15	4	1,7	11	60	18

Table 11: Duration of development projects calender months and person hours

	organization												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10a**	**10b**	**11**	**12**
Relative test cost (%)	-	35	10	15	33	33	65	33	35	65	45	40	12

Table 12: Relative cost of testing in development projects

	organization											
Test Phase	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Test plan	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Test specification	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Test status report	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Trouble report	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Test delivery note	No	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No
Final test report	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 13: Different types of documents produced in development projects

K Question 10 - Test Cases

Question 10 deals with test cases. On what grounds are test cases selected (table 17), are test cases reused from previous projects and in that case how? (Table 18) Other aspects of test cases also covered in question 10 are how test cases are stored (table 19) and what types of information that are contained in a test case (table 20).

L Question 11 - Metrics

Question 11 concerns which types of metrics the organizations collect and use in their projects (table 21).

	organizations
no test strategy	1,7,12
Implicit test strategy	2,5,6
Explicit test strategy, not used	9,11
Explicit test strategy, used	3,4,8,10

Table 14: Which organizations have and use test strategies. Numbers refer to organizations.

org.	Test strategy contents
2	General info on test case selection in test process.
3	Test team involvement, and improvement activities.
4	Required amount of testing and how to achieve this.
5	Automation strategy. Prioritize new functions in testing.
6	Requirements coverage, Standards for certain aspects of the product.
8	Which test activities should be executed. Test responsibilities for the different activities. Methods, tools, and trouble reporting, including classification of faults.
9	Different test phases, responsibility, checklists, parts of the test process.
10	Different test phases, responsibility. Also some standards from FDA and the European counterpart affect the test strategy.
11	Different test phases, responsibility, and which test documents each activity should produce.

Table 15: Contents of the test strategies that exist, implicitly or explicitly.

org.	Used Test Methods	Selection of Test Methods
4	EP, BVA, Combination Strategies	Test Strategy
8	EP, CEG, Own Methods	Testers & Managers choose
11	Req. coverage	Individual choice

Table 16: Test methods used by organizations and how test methods are selected.

M Question 12 - Cost

Finally, question 12 focuses on resource consumption. How is the total cost of testing distributed across the different test phases (table 22), and in system test, how much time is spent on planning, preparation, and execution? (Table 23)

N Subset of TPI Model

The Test Process Improvement (TPI) [KP99] method is used to rank the test maturity of an organization in twenty different key-areas. For each key-area $2-4$ maturity levels are defined, with requirements in order to reach that level and dependencies to other key-area levels. The organizations are ranked in the six key-areas in which there is enough information in this survey to be reasonably sure about the level. The details of these six key-areas can be found below

Table 24 shows the TPI ranking of the twelve organizations in this study for the six selected

org.	TC Selection
1	Testers' Experience
2	Testers' Experience and implicit strategy
3	Testers' Experience influenced by requirements coverage.
4	Test methods complemented with free testing in which testers use their experience within given directions.
5	Testers' Experience influenced by focus on new functionality.
6	Testers' Experience influenced by requirements coverage. Review of test cases by application engineers.
7	Testers' Experience influenced by requirements coverage.
8	Test methods complemented with testing of fixed faults and faults from operation.
9	Testers' Experience.
10	Testers' Experience.
11	Testers' Experience influence by a business perspective.
12	Testers' Experience.

Table 17: Selection of new test cases in a project.

key-areas. For each key-area, the number of levels are indicated by the letters after each key-area name. In each case "A" represents the lowest level, but in some cases it is possible to be less mature than required for the "A"-level. This is denoted with a dash.

The TPI ranks can be viewed as a summary of the results of this study.

O Details of subset of TPI Model

The following parts of the TPI model [KP99] were used when analyzing the data.

O.1 Test Strategy - key area 1

A Strategy for single high level test
Product risks are assessed. Different test depth are applied depending on risk. One or more test specification techniques are used to support the different test depths. Different test depths apply both to test and re-test.

org.	TC Reuse	On what grounds?
1	No	-
2	Yes	The amount of change in each system part.
3	Yes	Word of mouth from developers - what has changed.
4	Yes	Course-grained prio, but no specific strategy
5	Yes	All test cases that can be reused.
6	Yes	All test cases that increase requirements coverage.
7	Yes	All test cases that increase requirements coverage.
8	Yes	All test cases that can be used.
9	Yes	Based on testers experience.
10	Yes	Based on testers experience and requirement traceability.
11	Yes	As much as possible, based on testers experience.
12	Yes	As much as possible, based on testers experience.

Table 18: Reuse of test cases from old projects.

B Combined strategy for high-level tests
Coordination between different high-level tests (system, acceptance, production etc). Coordinated strategy put in writing. Coordinated strategy influences explicit test strategies for each level. Deviations from the test strategies are reported.

C Combined strategy for high-level tests and low-level tests or evaluation
Coordination between high-level tests and low-level tests or evaluation. Coordinated strategy put in writing. Coordinated strategy influences explicit test strategies for each level. Deviations from the test strategies are reported.

D Combined strategy for all test and evaluation levels
Coordination between high-level tests, low-level tests and evaluation. Coordinated strategy put in writing. Coordinated strategy influences explicit test strategies for each level. Deviations from the test strategies are reported.

O.2 Life-cycle Model - key area 2

A Planning, Specification, Execution
For the testing, at least the phases planning, specification, and execution are distinguishable.

B Planning, Preparation, Specification, Execution, Completion
For the testing, at least the phases planning, preparation, specification, execution, and completion are distinguishable.

org.	TC Storage
1	Do not keep test cases
2	Database
3	Paper based test specifications
4	Paper based test specifications
5	Paper based, version controlled, test specifications.
6	Paper based, version controlled, test specifications.
7	Paper based, version controlled, test specifications.
8	Database
9	Paper based test specifications
10	Paper based test specifications
11	Paper based, version controlled, test specifications.
12	Database

Table 19: Means of test case storage.

O.3 Moment of Involvement - key area 3

A Completion of test basis
The testing phase starts simultaneously with or earlier than the completion of the test basis (specifications).

B Start of test basis
The testing phase starts simultaneously with or earlier than the start of the test basis (specifications).

C Start of requirements definition
The testing phase starts simultaneously with or earlier than the phase in which the requirements are defined.

D Project initiation
When the project is started the activity testing is also started.

O.4 Test Specification Techniques - key area 5

A Informal techniques
Test cases are specified by means of a described technique. The technique requires the test cases to contain at least starting situation, action, and expected results.

Item	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Sensible Name	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Unique Id	N/A	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	indirect	No
Revision	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	indirect	Yes
Priority	N/A	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	No
Author	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Goal	N/A	Yes	No	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes
Purpose	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Configuration Reqs	N/A	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Precondition	N/A	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Action(s)	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Result(s)	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Postcondition	N/A	No	No	No	No	No	Yes	No	Yes	Yes	No	Yes
Type of Test Case	N/A	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	No
TC Passing Criteria	N/A	No	No	Yes	No	Yes	No	No	No	Yes	No	Yes

Table 20: Contents of stored test cases.

B Formal techniques
In addition to informal techniques, formal techniques are also used, providing unambiguous ways of getting from test basis to test cases. Test coverage relative to the test basis can be determined. The testware is reusable.

O.5 Metrics - key area 7

A Project metrics (product)
Project input metrics are recorded, e.g., used resources, performed activities, size, and complexity of tested system. Project output metrics are recorded e.g., test cases, test progress, and number of defects. Metrics are used in test reporting.

B Project metrics (process)
At least two of the following metrics should be monitored: Defect find-effectiveness, defect find-efficiency, test coverage level, testware defects, and perception of quality. Metrics are used in test reporting.

C System metrics
The metrics from level A and B are recorded also for development, maintenance and pro-

Type of Metric	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
used resources	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A
performed activities	N/A	No	Yes	Yes	No	Yes	Yes	No	No	Yes	Yes	N/A
size and complexity of tested system	N/A	Yes	Yes	Yes	No	No	Yes	No	No	No	No	N/A
test products	N/A	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	N/A
test progress	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	N/A
number of defects	N/A	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/A
defect find-effectiveness	N/A	Yes	No	Yes	No	No	No	No	Yes	No	No	N/A
defect find-efficiency	N/A	No	No	Yes	No	No	Yes	Yes	No	No	No	N/A
test coverage	N/A	No	Yes	Yes	No	Yes	Yes	No	No	Yes	No	N/A
testware defects	N/A	No	No	Yes	No	No	No	No	No	Yes	No	N/A
perception of quality	N/A	No	No	Yes	No	No	Yes	Yes	Yes	No	No	N/A
found faults in production	N/A	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	N/A
effectiveness of maintenance	N/A	No	No	No	No	Yes	No	No	No	No	Yes	N/A
efficiency of maintenance	N/A	No	No	No	No	Yes	No	No	No	No	Yes	N/A
collection and comparison of metrics across different products	N/A	No	No	Yes	No	No	No	Yes	No	Yes	Yes	N/A

Table 21: Types of collected and used metrics.

duction. Metrics are used in the assessment of the effectiveness and efficiency of the test process.

D Organization metrics (>1 system)
Organization-wide mutually comparable metrics are maintained for the already mentioned data. Metrics are used in assessing the effectiveness and efficiency of the separated test processes, to achieve an optimization of the generic test methodology and future test process.

O.6 Test Functions and Training - key area 12

A Test managers and testers
The test personnel consists of at least a test manager and some testers. The tasks and responsibilities are defined. The test personnel has specific test training. For the acceptance testing, expertise in the subject matter is available to the testers.

Test Phase	organization								
	3	**4**	**5**	**6**	**7**	**8**	**9**	**11**	**12**
Pre-execution (%)	10	35	0	10	5	40	10	20	0
component (%)	0	5	5	50	30	0	10	10	0
integration (%)	50	0	30	0	0	10	0	0	20
system (%)	25	40	60	40	60	35	10	60	80
acceptance (%)	0	10	5	0	5	15	70	10	0

Table 22: Relative amount of testing across different test phases. Orgs 3 and 4 have some field testing in addition to the reported data.

	organization												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10a**	**10b**	**11**	**12**
Planning (%)	5	20	6	10	5	10	10	30	20	10	10	10	20
Preparation (%)	10	40	28	50	15	60	20	30	60	10	70	50	20
Execution (%)	85	40	66	40	80	30	70	40	20	80	20	40	60

Table 23: Relative amount of time spent in system testing on planning, preparation and execution.

B Methodical, Technical, and Functional support of test process, testware, and infrastructure
Methodical support is a separate activity. Technical support is a separate activity. Functional support is a separate activity. Management of test process is a separate activity. Management of testware is a separate activity. Management of test infrastructure is a separate activity. The responsible persons have sufficient knowledge. Time for these activities is planned.

C Formal internal quality assurance
An internal QA plan for testing is formulated. The QA person has no other tasks in the test team. The results of the QA activities are used for further improvement of the test process. The QA person has sufficient knowledge and experience.

O.7 Unused key areas

The following key areas (number and name) were unused in this study.

4 Estimating and Planning

6 Static Test Techniques

8 Test Tools

key-area	organization											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1. Test Strategy (A-D)	-	-	-	B	-	-	A	B	A	C	-	-
2. Life-cycle Model (A-B)	-	A	A	A	-	A	A	B	A	A	-	-
3. Moment of Involvement (A-D)	A	C	C	D	A	C	D	C	D	D	D	D
5. Test Specification Techniques (A-B)	-	-	A	B	-	A	A	B	A	A	-	-
7. Metrics (A-D)	-	A	A	B	-	A	B	A	A	A	-	-
12. Test Functions and Training (A-C)	-	-	A	B	-	A	B	A	A	A	-	-

Table 24: Partial TPI assessment of the surveyed companies

9 Test Environment

10 Office Environment

11 Commitment and Motivation

13 Scope of Methodology

14 Communication

15 Reporting

16 Defect Management

17 Testware Management

18 Test Process Management

19 Evaluation

20 Low-level Testing