UNIVERSITY OF OSLO
Department of Informatics

How do software practitioners value research when making decisions?

Master thesis

Kjetil Ree

August 3, 2009

Abstract

Because researchers in empirical software engineering need to better understand software practitioners' attitudes to research, we designed a personal opinion survey and deployed it in two Norwegian software houses and a Norwegian Java users' group.

Statistical analysis using the software package *JMP* showed that most practitioners overwhelmingly rely on colleagues and friends when learning about and considering implementing new technologies. They value the advice of experts over research, and do not differentiate between industrial and independent (academic) research. Practitioners rely almost equally on their own analysis, experts' advice and intuition when making important decisions.

A majority of software practitioners claim to apply research in software engineering in their work. Neither the age, the education, nor the industry experience of practitioners significantly influenced their attitudes to research. Being too busy meeting immediate goals / deadlines and lack of personal time were the two most widely reported barriers to applying research, while neither organizational inertia, attitudes of colleagues or customers, nor the relevance of research were seen as barriers.

Topics related to management were mentioned most often by practitioners when asked about which topics researchers should focus more on.

Acknowledgements

I would like to express my gratitude to my supervisor Jo Hannay. Without his understanding and patience, this thesis would never have been completed. I would also like to thank Magne Jørgensen and Stein Grimstad for valuable advice and feedback.

On extremely short notice, Trygve Laugstøl of javaBin provided invaluable help with assisting me in deploying my survey. I owe my sincere thanks to javaBin and all my anonymous respondents.

I would also like to thank both students and employees at Simula Research Laboratory for a great social environment.

Oslo, August 2009
Kjetil Ree

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

One of the current buzzwords in SE research is *evidence–based software engineering* (EBSE). This paradigm was proposed by Dybå, Kitchenham and Jørgensen, and is inspired by *evidence–based medicine*, the current practice in medical research [8]. The goal of EBSE is that practitioners should improve decision making in software development and maintenance by integrating current best evidence in research with practical experience and human values [29].

An obvious goal of research in empirical software engineering is that it should be of interest to software practitioners, including both developers and managers. This is sometimes the case, but in other situations, software practitioners say that research in empirical software engineering does not address the issues that concern them, or even indicate that research in empirical software engineering is of no interest at all.

If evidence–based software engineering is to succeed, it is critical that practitioners actually care about research. Other issues are also important (for instance that research is applicable and relevant to the problems software practitioners encounter), but these issues are of secondary significance if nobody uses research. If practitioners do not care about research at all, the proponents of EBSE will have a very long way to go in order to make any impact with this paradigm.

1.2 Objective

In this thesis, we look into how software practitioners make decisions, and whether research influences these decisions. The objective of this thesis is to elicit practitioners' attitudes to research, as researchers may work in vain if industry does not care about their results.

In order to address these issues, we conducted a survey among software practitioners. The data collected in this survey was used for answering the following research question:

RQ 1 *How do software practitioners value research when making decisions, hereunder:*

SRQ 1 *How do software practitioners learn about new techniques and technologies?*

SRQ 2 *Do software practitioners value experts' advice more than they value research?*

SRQ 3 *Which obstacles exist for software practitioners to use research?*

SRQ 4 *Which topics do software practitioners feel that researchers should focus on?*

1.3 Research Method

As described in chapter 4, we conducted a survey to elicit opinions from industry practitioners. We deployed the survey in two Scandinavian consultant companies, and in a Norwegian Java users' group.

The data collected from 113 respondents was later analyzed quantitatively using *JMP*, a statistical software package. We also did exploratory modeling, employing a statistical technique called *recursive partitioning* used for finding statistically significant relationships between variables.

1.4 Main Contributions

A main finding is that ICT practitioners overwhelmingly rely on colleagues and friends when learning about and considering implementing new technologies. Most practitioners neither consult nor rely on researchers when making decisions about things they have little prior knowledge of or when they are learning about new technologies; but practitioners rely almost equally on their own analysis, intuition, and experts' advice when making important decisions.

A majority of practitioners claim to apply research in software engineering in their work. Being too busy meeting immediate goals / deadlines and lack of personal time were the two most widely reported barriers to applying research. Many practitioners feel that researchers should focus more on topics related to management.

1.5 Outline

In chapter 2, we give a quick introduction to empirical software engineering, and then describe its industrial application, evidence-based software engineering. We discuss why practitioners' attitudes are important, and also look at attitudes in other industries.

Chapter 3 summarizes some prior research that is relevant to our purposes. We discuss a study that look at which research topics are popular in software engineering, and also discuss a research project that aims to determine the impact of software engineering research on software engineering practice. We also look at three classic research papers in the domain of cognitive science, and describe why these are important to the understanding of experts and intuition.

Chapter 4 starts by briefly discussing some research methods in empirical software engineering, and then gives reasons for why a personal opinion survey is suitable for our purposes. We discuss some key aspects of surveys, and present a survey specification that in an annotated form gives all the questions and their rationales. We then discuss the evaluation process, and how we deployed the survey. The chapter ends with a discussion on data quality.

In chapter 5, we present the results. We start by discussing why we treat our data as being on continuous scales, briefly discuss some key statistical concepts, and then show the distributions of the answers to the 10 questions and a summary of the demographic data. We then describe the statistical technique of recursive partitioning, and report 12 scenarios where we have applied this technique to find statistically significant relationships between variables.

In Chapter 6, we summarize the results and discuss our findings. We also look at some possible validity threats, and briefly discuss research ethics.

Chapter 7 gives a short summary of our goal, method, and contributions. We also give some ideas for future work.

Chapter 2

Background

This chapter gives a short description of *empirical software engineering* in general (section 2.1) and its industrial application *evidence–based software engineering* more specifically (section 2.2). Section 2.3 discusses why practitioners' attitudes to research is important if they are to apply evidence–based approaches to software engineering. Section 2.4 shows some findings from a survey about medical practitioners' attitude to the evidence–based paradigm, and section 2.5 quickly looks into the rationality of industry.

2.1 Empirical software engineering

The world is increasingly becoming more computerized, in a scale that few could imagine only 20 years ago. Information systems are prevalent in all branches of society, and software and technology are driving forces in the modernization of the world we live in.

The exponential growth in hardware capabilities starting in the late 1960s made it for the first time possible to create major software systems. This presented a problem to software practitioners, as the previously employed processes did not scale up well for more complex systems. This coined the term "software crisis", as it was a widespread notion that software had become expensive, unreliable, unmaintainable, and underperforming [7].

The "software crisis" forced industry and the research community to view software development as something resembling traditional engineering processes, leading to a major development of software process models and creating the new discipline of software engineering (SE). As defined by Sjøberg et al., software engineering is about [44]

> developing, maintaining, managing high–quality software systems in a cost–effective and predicable way,

while research in software engineering concerns

> (1) the development of new, or modification of existing, technologies (process models, methods, techniques, tools or languages) to support SE activities, and
> (2) the evaluation and comparison of the effect of using such technology in the often very complex interaction of individuals, teams, projects and organisations, and various types of task and software systems.

Sjøberg et al. consider activities (1) and (2) mutually dependent, and notes that "[s]ciences that study real–world phenomena, i.e. empirical sciences, of necessity use empirical methods [. . .] if SE research is to be scientific, it too must use empirical methods".

Perry et al. give the following description of an empirical study [37]:

> [. . .] the essence of an empirical study is the attempt to learn something useful by comparing theory to reality and to improve our theories as a result. Therefore, empirical studies involves the following steps:
>
> - formulating an hypothesis or question to test,
> - observing a situation,
> - abstracting observations into data,
> - analyzing the data, and,
> - drawing conclusions with respect to the tested hypothesis.
>
> Of these, the last step — drawing conclusions – is the most important and too often the least well done.

The combination of software engineering research and empirical methods led to the term *empirical software engineering* (ESE). According to Sjøberg et al., an empirical approach to software engineering started on a large scale in the 1970s, and there has later been an increased focus on the need for applying empirical methods in software engineering research. Despite of that, one is far away from their vision, which is that "[. . .] scientific knowledge should guide the development of new SE technology and be a major input to important SE decisions in industry and services."

2.2 Evidence–based software engineering

Empirical software engineering has been applied to industrial needs by Kitchenham, Dybå, and Jørgensen [29]. Their approach, known as *evidence–based software engineering* (EBSE), is inspired by the evidence-based paradigm of medical research. Evidence–based medicine developed in the late 1980s and the 1990s, when it became apparent that systematic reviews were superior to the judgement of experts, where the latter practice could result in medical failures and ultimately cost lives.

The success of evidence–based practices in medical research has prompted such practices to be adopted in many other fields of research, including psychiatry, nursing, social policy, and education. This has lead Kitchenham et al. to sugest a similar practice in software engineering, not just because "everyone else is doing it", but because they believe that "a successful innovation in a discipline that, like software engineering, attempts to harness scientific advances for the benefit of society, is worth investigating" [29].

There are five basic steps of evidence–based software engineering [8]:

1. Convert a relevant problem or information need into an answerable question.
2. Search the literature for the best available evidence to answer the question.
3. Critically appraise the evidence for its validity, impact, and applicability.
4. Integrate the appraised evidence with practical experience and the customer's values and circumstances to make decisions about practice.
5. Evaluate performance and seek ways to improve it.

This procedure seeks to integrate current best evidence with practical experience and human values, something which hopefully will lead to improved decision making related to software development and maintenance. Dybå, Kitchenham, and Jørgensen do not however expect technologies to be universally good or universally bad, but only "more appropriate in some circumstances and for some organizations." They also stress the need for practitioners who use EBSE to "accumulate empirical research about a technology of interest and evaluate the research from from the viewpoint of their specific circumstances" [8].

Dybå, Kitchenham, and Jørgensen point out several pitfalls for practitioners who would like to use an evidence–based approach. One of these is that it could be hard for practitioners to critically appraise evidence. This could be countered if the scientific community published more systematic reviews — one of the most convincing forms of evidence [8]. Reviews can help practitioners to identify faulty research, and there are several notable examples of such. One example is the GAO Study, a report by the U.S. Government Accounting Office that described a terrible failure rate among studied software projects. It was later found that the much cited GAO Study *was a study of projects known to be failing*, and when this was discovered, the study was quickly dropped as a citation to support the notion of a "software crisis" [17]. Another, perhaps even more problematic study, is the Standish Group's 1994 *Chaos Report*. This report is one of the most widely cited statistics in the IT industry. Jørgensen and Moløkken-Østvold have compared the numbers of the Chaos Report with the findings of other studies [26]. While the Chaos Report spoke of a "189% average cost overrun", a systematic search for other estimation studies showed that these other studies found the average cost overrun to be 33–34%. Even though the numbers were not directly comparable, there was no way of explaining the huge difference between the other studies and the Chaos Report. Jørgensen and Moløkken-Østvold also evaluated the design of the Standish Group's study, and found the design to be poorly described, in particular how "cost overrun" was defined. When they contacted the Standish Group asking for an explanation, they got the reply that "providing this type of information would be like giving away their business free of charge", and no clarification of how "cost overrun" was defined. The incomplete description of the study and the lack of a definition of the key concept "software overrun" make it hard for both researchers and practitioners alike to evaluate its validity, and thus makes the Chaos Report unsuitable for an evidence–based approach.

2.3 The need for researchers to examine practitioners' attitudes to research

There are some conditions that need to apply if evidence–based software engineering is to become a widespread and useful practice. These include (but are not limited to) that (1) practitioners need to have faith in the value of proper scientific methods, (2) research must be relevant and applicable to industry's needs, (3) practitioners must know how to find and appraise relevant research, and (4) the EBSE approach has to prove itself useful — it must yield good results when it comes to developing, maintaining, and managing software systems.

We spent a significant amount of time searching scientific databases for literature discussing software practitioners' attitudes to research. We mainly searched in *IEEE Xplore* (`http://ieeexplore.ieee.org`) and in *ACM Digital Library* (`http://portal.acm.org/`), and also to some extent in *Google Scholar* (`http://scholar.google.com`). IEEE and ACM were chosen because they are widely regarded as the most prestigious publishers, and Google Scholar was chosen because of its breadth.

Our searches were mostly fruitless, and it appears that there has been published very little or no research that addresses these issues. While all of the issues are important and worthy of looking at, time constraints force our thesis to only focus on some of them. As it was necessary to narrow down the research question (RQ) in order to make an answerable question, we chose to add a "when making decisions" part to a variation of (1), leading to RQ 1 — "how do software practitioners value research when making decisions?"

We also included more specific research questions (SRQs), these do to some extent cover (2) and (3). It was a natural choice to contrast research to experts' advice (SRQ 2), as experts often make strong claims that they have little or no empirical evidence to support. One example is agile hardliner Dave Astels, who claims that projects he knows that use Test Driven Development "without exception" experience "dramatic increases in quality and significant overall time savings" [15]. As far as we can see, this claim is not supported by research. For example, in a summary of empirical studies of TDD by Erdogmus et al., two of the three controlled experiments cited did not find any difference between the TDD group and the control group when it came to productivity, while the third experiment reported that TDD yields worse results [10].

2.4 Attitudes in other industries

As there has been no previous SE research that addresses these issues, it is useful to take a look at similar research in other disciplines that can give an understanding of why these issues are important. As noted in section 2.2, evidence–based software engineering has its origins in evidence–based medicine. There have been several studies regarding medical practitioners' attitudes towards the evidence–based paradigm. Among these are McColl et al., who conducted a questionnaire survey to determine the attitude of general practitioners towards evidence–based medicine and their related educational needs [33]. They asked practitioners about

(A) their attitude towards current promotion of evidence based medicine,
(B) perceived attitude of colleagues towards evidence based medicine,
(C) if practicing evidence–based medicine improved patient care,
(D) the practitioners' perceived usefulness of evidence based medicine in day to day management of patients, and
(E) the estimated percentage of the respondent's clinical practice that is evidence based.

On a scale where 100 was "strongly agree" (C) and "extremely useful" (D) and 0 was "strongly disagree" (C) and "totally useless" (D), the median value for (C) was 70 and the median value for (D) was 63. The median value for (E) was 50%. While a clear majority of the respondents found evidence–based practices useful, only 50% of the respondents' actual clinical practices were based upon the evidence–based approach. McColl et al. also examined the practitioners' reasons for not applying evidence–based practices. The main barrier was "lack of personal time" (171 of 242 participants), while other major barriers were "personal and organisational inertia" (35 of 242), "attitudes of colleagues" (29 of 242), "patients' expectations" (23 of 242), "lack of hard evidence" (20 of 242), "evidence not related to context of primary care" (16 of 242), and "availability and access to information" (14 of 242). (Respondents gave more than one answer.)

These findings are for many reasons not directly applicable to EBSE. A major reason is that evidence–based medicine is a mature and well known practice, while evidence–based software engineering still is in its infancy and probably mostly unknown to the great majority of practitioners and researchers. This is indicated by the fact that Google Scholar can find about 114.000 scientific articles mentioning the term "evidence–based medicine", but only 259 scientific articles that mention the term "evidence–based software engineering". The huge difference in the impacts of evidence–based practices in these two disciplines makes it impossible to use the number of medical practitioners that find evidence–based practices useful as in indicator of how many software practitioners that have similar attitudes — we assume that if one asks software practitioners whether practicing evidence–based software engineering improves software maintenance, very few software practitioners would be able to give an answer.

There are however some interesting aspects of McColl et al.'s findings. Several of the barriers reported by the medical practitioners have their equivalents in software engineering. "Lack of personal time" is by far the most widely reported barrier by medical practitioners, and it is certainly reasonable to expect this barrier to be prevalent also in the software community. It is possible that practitioners simply do not have the time to stay updated on research. Some of the other barriers, such as "personal and organisational inertia" and "attitudes of colleagues", are also likely applicable in software milieux. The same can be said about "patients' (customers') expectations" — if customers want a particular strategy or solution, it could be quite hard for developers to persuade them that research shows that their wishes are unfavorable.

2.5 The industry and rationality

It is not always obvious that managers and practitioners make rational decisions based upon research. One example is the impact of agile practices. Dybå and Dingsøyr cite studies that show that half of companies in Europe and the United Sates are considering a switch to agile methods, and note that the "Agile Conference" has grown to be one of the largest software engineering conferences in just six years [9]. However, the large interest in agile practices is not necessarily backed up by research. Dybå and Dingsøyr conducted a review of published studies to summarize what currently known about the benefits and limitations of agile software development. They found that "the strength of the evidence in the current review regarding the benefits and limitations of agile methods, and for decisions related to their adoption, is *very low*", and that "any estimate of effect that is based on evidence of agile software development from current research is very uncertain." Agile management methods, such as Scrum, particularly need more attention.

Dybå and Dingsøyr's review raises several important questions, as industry's enthusiasm for switching to agile practices can seem irrational when one view their findings. If virtually all studies of agile methods are based upon very weak evidence, why is industry so eager to adopt such methods? Could it be that industry does not care about research, but rather choosees to have faith in agile evangelists? Or could it be that they want to use research, but are unable to understand the "researcher lingo" or to assess the validity of research? It is important that researchers get feedback on such questions, something that SRQ 2 will provide.

2.6 A silver bullet?

The term "No Silver Bullet" was coined by Fred Brooks in his influential 1986 essay *No Silver Bullet — Essence and Accident in Software Engineering*. In his essay, Brooks claimed that [3, p. 179]

> [t]here is no single development, in either technology or management technique, which by itself promises even one order–of–magnitude improvement within a decade in productivity, in reliability, in simplicity.

Inspired by Aristotle, Brooks divided difficulties in software engineering into *essence* and *accidents*. Brooks defined the *essence* of software as a "construct of interlocking concepts: data sets, relationships among data items, algorithms, and invocations of functions," while *accidents* are "those difficulties that today attend its production but are not inherent" [3, p. 182]. He believed the specification, design, and testing of this construct to be the hard part, not the "labor of representing it and testing the fidelity of the representation."

As Brooks considers conceptual errors the most severe, he claims that building software will always be hard, and that there is inherently no silver bullet. His claims have however been challenged by others, among them Brad Cox. In 1995, Cox claimed that there indeed is a silver bullet, and that [5, p. 378]

> [t]he silver bullet is a *cultural* change rather than a technological change. It is a paradigm shift — a software industrial revolution based on reusable and interchangeable parts that will alter the software universe as surely as the industrial revolution changed manufacturing.

Nearly 15 years after Cox' bold claims, component–based software engineering (CBSE) has far from revolutionized the discipline of software engineering. The open market for components has not yet developed, and serious problems like component trustworthiness, component certification and emergent property prediction remain [45, p. 441].

The proponents of evidence–based software engineering have wisely avoided making too strong claims about its possible impact. As discussed in section 2.2, the goal of EBSE is merely to "improve decision making related to software development and maintenance" by emphasizing methodological rigor [8], and it is thus not intended to be a "silver bullet."

In our opinion, EBSE concerns the "construct of interlocking concepts"; what Brooks called the *essence* of software engineering. The *New Oxford American Dictionary* defines the noun *construct* as "an idea or theory containing conceptual elements, typically one considered to be subjective and not based on empirical evidence." A change to an evidence–based approach, which by design is founded on the use of objective and empirical evidence, will hence be a revolution if it is viable.

Predictions like the one by Cox have without exemption turned out to be incorrect, so we will certainly refrain from claiming EBSE to be a silver bullet. However, like Kitchenham, Dybå, and Jørgensen, we think the idea is worth investigating. Our RQ 1 ("how do software practitioners value research when making decisions") is derived directly from the goal of EBSE.

2.7 Summary

We have seen how the perceived "software crisis" lead to the growth of software engineering (SE) as a new engineering discipline, and how this was combined with empirical methods to form empirical software engineering (ESE). The industrial application of empirical software engineering, evidence–based software engineering (EBSE), was introduced in the 2000s and based upon evidence–based medicine.

If EBSE is to succeed, it is critical that researchers gain knowledge about practitioners' attitudes to research. This thesis will look at this issue, more specifically at how practitioners value research when making decisions.

We have seen that there has not been any published software engineering research concerning this problem, and therefore had a quick glance at attitudes in medicine, a "hard science" in many ways related to engineering and other physical sciences. A majority of medical practitioners found an evidence–based approach useful, but there were also many barriers to using research in their daily work. These barriers could very well also exist among software practitioners. We have also seen an example of how the software industry embraces new methods even though research does not necessarily show clear benefits.

This chapter has shown that there is is a clear need for the research community to better understand practitioners' attitudes to research when making decisions. In the next chapter, we will look at some relevant prior work that will help us design and analyze a survey that can give us insight into this issue.

Chapter 3

Related work

In chapter 2, we looked at the background of evidence–based software engineering, and why it is important for researchers to gain insight into how practitioners value research.

This chapter summarizes some of the related work — that is, previous research that is related to what we are doing and that it is important to take into consideration when we design our survey and analyze our data.

Section 3.1 summarizes the work of Cai and Card, who have looked into which topics researchers in software engineering focus on. This is important to take into consideration when we analyze which topics software practitioners feel that researchers should focus on (SRQ 4), as we can compare practitioners' wishes to the actual production of journal and conference papers in the field of software engineering.

In section 3.2, we look at how a project called *The Impact Project* has assessed the impact of research in software engineering. We also see that it could take 10–20 years for research to reach industrial applications, something which could make practitioners feel that research is "irrelevant" and "outdated", suggesting that they do not learn about new techniques and technologies from research.

Section 3.3 enters the realm of cognitive science ("the nature of intelligence"). In this section, we look at the work at some of the world's most influential cognitive psychologists. The studies cited in the three following subsections look into how intuition often is wrong and why experts could be overconfident, something which should make an analytical approach based upon research more beneficial to practitioners (RQ 1, SRQ 2).

3.1 Cai and Card

Cai and Card have looked into what the active research focuses are within the field of software engineering [4]. This main question further raised two subquestions, namely "what are the popular research topics in the field of software engineering", and "what are the evolving trends of these research topics".

To gather data for analyzing these issues, Cai and Card selected a number of software journals based upon their impact factor and cited half-life. They selected journals covering a broad range of topics, so that the results would not be biased. Their selection included a total of seven journals. They also included seven top international conferences, more or less subjectively chosen.

To classify the research papers from these seven journals and seven conferences, Cai and Card used the ACM Computing Classification System, a classification system that divides SE research topics into classes such as "design tools and techniques", "testing and debugging", "metrics", and so on.

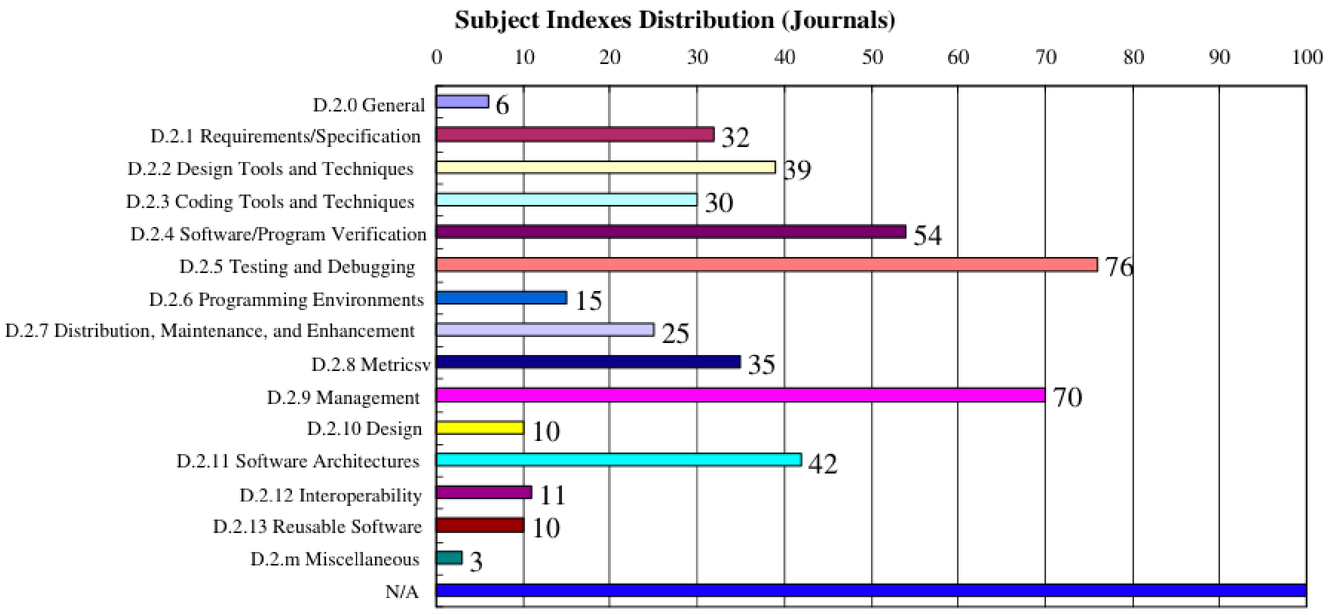
From the seven top journals and seven conferences, Cai and Card examined all papers manually to identify their research topics and classify them. Each paper could be put into one or more of the classes, or alternatively put into a special "N/A" class if none of the ACM Computing Classification System classes were suitable. The classification was done by using the papers' keywords, and by using the abstract or full text if no keywords were supplied. Some articles (such as the opinion items and column comments of the *IEEE Software* magazine) were not counted, as these are not really research articles. For the seven conferences, papers that were only presented during satellite workshops were excluded.

Cai and Card's distribution of subjects from the examined journals is shown in figure 3.1(a). The largest by far is the "N/A" class (123 papers). Among the topics proper, "testing and debugging" and "management" are the largest with a count of 76 and 70, respectively. Other topics that are popular are "software / program verification" (54), "software architecture" (42), "design tools and techniques" (39), and "metrics" (35). Among the topics that are less discussed in journals are "miscellaneous" (3), "general" (6), "design" (10), "reusable software" (10), "interoperability" (11), and "programming environments" (15).

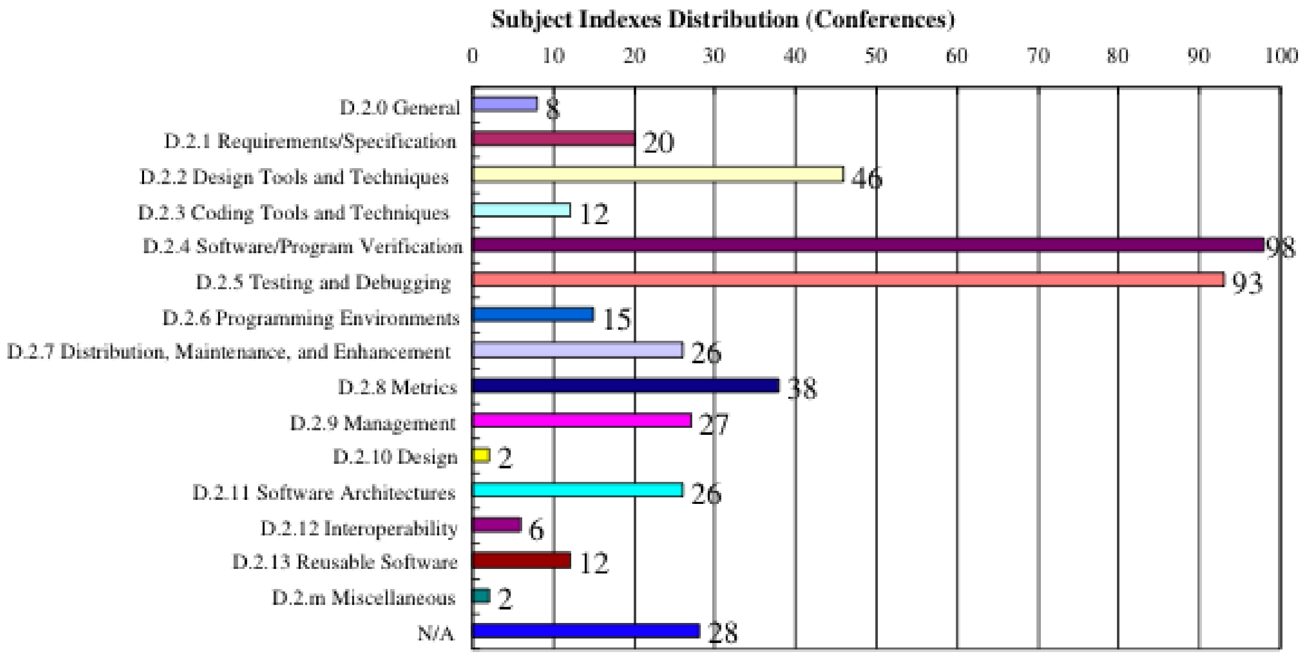
There were considerable differences between the distribution of topics from conference papers and the distribution from conference papers (see figure 3.1(b) for the latter). The most popular topic of conference papers was "software / program verification" (98 papers), while "testing and debugging" came second (93). The least popular topics were "miscellaneous" (2), "design" (2), "interoperability" (6), "general" (8), "coding tools and techniques" (12), and "reusable software" (12).

Combining the distribution of topics from conference papers and journal papers, it is clear that some software engineering topics are vastly more popular than others. The overall most popular topics are "testing and debugging" (169), "software / program verification" (152), "management" (97), and "design tools and techniques" (85). Other important topics such as "requirements / specifications" (52), "distribution, maintenance, and enhancement" (51), "programming environments" (30), and "reusable software" (22) were clearly less popular.

Cai and Card's findings are interesting. There is for example published almost seven times as many SE research papers on the topic of program verification as there is on software reuse. In spite of that, it is not at all obvious that this is what the software



(a) Journal papers



(b) Conference papers

Figure 3.1: Excerpt form Cai and Card's findings: The distribution of subject indexes from journal and conference papers [4].

industry needs. Ian Sommerville notes in his widely used textbook *Software Engineering* that "whether or not to reuse is often a managerial rather than a technical issue" because of the wide array of reuse techniques available, and that managers "may not understand the risk associated with reuse as well as they understand the risk of original development" [45, p. 421]. One can safely assume that research on the topic of software reuse could be very interesting to the software industry, even though this topic seems to get little priority from researchers. Similar cases can be made for many of the other topics, there is no obvious correlation between what one can presume that industry wants and what researchers publish. This underlines the need for researchers to examine if their focuses are in line with the requirements of the software industry, and is closely associated with our SRQ 4.

3.2 Osterweil et al. and *The Impact Project*

A research project called *The Impact Project* strives to determine what impact, if any, SE research has had on SE practice. The project aims to be highly inclusive, involving the efforts in all areas of software engineering. The project does not only include academic researchers, but also industrial researchers and a wide range of practitioners. However, limited resources have made it a necessity for The Impact Project to only select some representative areas, such as software configuration management, inspections, reviews and walkthroughs, middleware, and runtime assertion checking.

Some of the project's preliminary findings and the project's general organization and research methods have been summarized by Osterweil et al. [36]. In each area, the project is compiling one or more reports, each report being authored by six to eight academic and industrial experts. The approach in each area is to start at a practice and work backward in time to find the practice's roots, and they have deliberately avoided to start at the research and working forward in time to get to the practices, as the latter procedure may be seen as an attempt to justify individual research ideas rather than to find the genuine impact.

Osterweil et al. claim that even though the perception that research does not have an impact on practice seems widespread, the early findings of the Impact Project suggest that there is a substantial impact. Their main claim is that software engineering research has significantly and positively affected software engineering practice. They note that the "successful deployment of technologies that support practice clearly requires many types of contributions from many types of participants", while the studies they cite supposedly "make it clear that researchers' participation has been of considerable importance" (even though they are very vague when it comes to the details of these studies).

Another of the main claims by Osterweil et al. is that "lasting impact seems to come most readily from ongoing interactions between research and practice". They claim that while technology transition could be thought of as moving a key idea or a prototype out of the research environment, their studies indicate that "success in practice seems to require continued interactions between research and practice, resulting in continued upgrading and improvement of a capability".

Another important finding by Osterweil et al. is that "research impact might not be fully felt for at least 10 years", and that it "typically takes 10 to 20 years for a good idea to move from initial research to final widespread practice". These numbers are in line with research from the mid 1980s. A number of case studies made by Redwine and Riddle show that the average time from the emergence of a key idea until there was substantial evidence of external value and applicability was 17 years, with a worst case of 23 years and a best case of 11 years [38]. Redwine and Riddle do however note that these times can be considerably shorter than the average in some special cases, such as for example methodology technologies that can transfer quickly to a homogenous target community if there is a clear need.

The long time frame from when basic research is conducted until the results are applicable in industry (10-20 years) may lead to negative attitudes from practitioners, as they may have a hard time seeing the connection between what they do and what researchers find. If practitioners thus perceive research as "irrelevant" or "outdated", it is reasonable to believe that they do not learn about new techniques and technologies from research (SRQ 1). It could also be that practitioners believe that certain topics are under–researched because it takes many years until recently invented ideas from these topics are useful to them, thus making practitioners feel that relatively "hot" topics that already are popular among researchers are under–researched (SRQ 4).

3.3 Experts and intuition : always right?

Evidence from studies in cognitive science suggests that experts and intuition are not always trustworthy. The following three subsections summarize three papers that shed light on these issues.

The concepts of *attribute substitution*, *substitution of representativeness for probability*, and *neglect of base rates* are described in section 3.3.1. The examples given by Kahneman and Frederick show how attribute substitution makes intuition unreliable, because people who are asked difficult questions often unconsciously answer an easier, similar question. Attribute substitution is also something one needs to have in mind when designing a survey. A similar problem is that people often substitute representativeness for probability. Kahneman and Frederick use the famous "Linda" problem to show that reliance on representativeness make people violate basic rules of logics. Substitution of representativeness for probability and neglect of base rates are concepts which show that using intuition will yield inferior results in comparison to careful analysis.

In a series of studies by Griffin and Tversky (section 3.3.2), they show evidence for their hypothesis that experts are more prone to overconfidence than ordinary people when predictability is very low. They also show that people make decisions based upon beliefs about individual events rather than about overall base rates, something that makes intuitive judgment depart from normative theory. The strong evidence for how intuition departs from normative theory should in an ideal world shift practitioners away from intuition and towards a more analytical approach (i.e. allocating more value to research when making decisions). In a similar fashion, the overconfidence of experts should discourage practitioners from valuing experts' advice too much.

Tversky and Kahneman (section 3.3.3) describe how people have incorrect assumptions about the laws of chance; what they call "the law of small numbers." In their experiment, mathematicians and psychologists were given several scenarios and were asked some questions about statistics and sampling related to these scenarios. These groups are clearly acquainted with logics and statistical theory, but they nevertheless believed that samples drawn form a population are more similar to another than sampling theory predicts. This finding has implications for our research question: Software practitioners should be acquainted with logics (either formally — for example through college courses in predicate logic — or more informally, for example through practical experience with algorithms) and to some degree also with statistics, but will still make mistakes when making assumptions about the laws of chance.

The work of Tversky and Kahneman and their associates shows that the human mind is riddled with cognitive biases, something which makes intuition unreliable. As this also applies to experts, it should be rational to value rational analysis over intuition and experts' advice when making important decisions. This is important when discussing and analyzing SRQ2.

3.3.1 Kahneman and Frederick : Attribute substitution and substitution of representativeness for probability

Kahneman and Frederick describe something called *attribute substitution* [27]. This phenomenon makes people who are asked a difficult question answer an easier question, often without the person being aware of the change. For example, a person who is asked "What proportion of long–distance relationships break up within a year?" may answer as if he had been asked "Do instances of failed long–distance relationships come readily to mind?". According to Kahneman and Frederick, "attribute substitution occurs when a relatively inaccessible target attribute is assessed by mapping a relatively accessible and related heuristic attribute onto the target scale."

Attribute substitution often make intuition unreliable. Kahneman and Frederick cite an experiment where the subjects were given the following question:

> If a sphere were dropped into a open cube, such that it just fit (the diameter of the sphere is the same as the interior width of the cube), what proportion of the volume of the cube would the sphere occupy?

The subjects' mean estimate was 74%. This figure is scarcely different form the mean estimate of a similar problem, namely "If a *circle* were drawn inside a *square*, what proportion of the *area* of the square does the circle occupy?" (77%). The correct answer of the sphere/cube/volume problem is 52%, way lower than the mean estimate. This strongly indicates that respondents answer as if they were asked the simpler, "two–dimensional" question. Kahneman and Frederick note that when "the target attribute in this judgment (the volumetric relation between a cube and a sphere) is simple enough to be understood but complicated enough to accommodate a wide range of estimates as plausible answers" while "a relevant simpler computation or perceptual impression exists", the respondents "will have no strong basis for rejecting it as their 'final answer'."

Kahneman and Frederick also uses the example of the puzzle "a bat and a ball cost $1.10 in total. The bat costs $1 more than the ball. How much does the ball cost?". Almost everyone will answer "10 cents", as $1.10 split nicely in $1 and 10 cents, and 10 cents is in the right magnitude. 10 cents is of course incorrect, but even half of the undergraduates at elite institutions make this mistake. And as the puzzle is not really that hard (nor ambiguous), it is clear that the people who make the mistake did not take the trouble to check their own answers. To explain the many mistakes with the "bat and ball problem", Kahneman and Frederick put forward the argument that "people often make quick intuitive judgements to which they are not deeply committed."

Similar concepts are the *substitution of of representativeness for probability* and the *neglect of known base rates*. Kahneman and Frederick exemplify these concepts with the famous *Linda problem*, first described by Tversky and Kahneman in 1982 [47]. A woman named Linda was described as follows:

> Linda is 31 years old, single, outspoken and very bright. She majored in philosophy. As a student she was deeply concerned with issues of discrimination and social injustice and also participated in antinuclear demonstrations.

Two separate groups of respondents were asked to rank a set of eight outcomes by representativeness and by probability. Six of these eight outcomes were fillers ("elementary school teacher", "psychiatric social worker" etc.) while outcome #6 was "bank teller" and outcome #8 was "feminist and bank teller".

There was an almost perfect correlation (.99) between how the first group of respondents ranked representativeness and how the second group ranked probability. The most interesting part is that most of the respondents ranked the conjunction ("feminist and bank teller") *higher* than its constituent ("bank teller"), *both* in representativeness (85%) and probability (89%). It is reasonable to say that Linda resembles a feminist bank teller more than she resembles a bank teller, but she can not be more likely to be a *bank teller and a feminist* than being *just a bank teller*! Kahneman and Frederick thus notes that "reliance on representativeness yield probability judgments that violate a basic logical rule."

3.3.2 Griffin and Tversky : Evidence and Confidence

In a series of experiments, Griffin and Tversky looked into how people weigh evidence and determine their confidence in a hypothesis by balancing arguments for and against it [19]. One of their findings is that people are better at predicting their own behavior than predicting others. In one of their experiments, they made fourteen pairs of same–sex students play a "Prisoner's Dilemma"–type game where in a series of trials, the students had to choose between cooperating and competing with the partner. There were 20 possible strategies for choosing between cooperation and competition, where some were designed to encourage cooperating, and others designed to encourage competition. The subjects were instructed to make predictions for the strategies, in addition to stating their confidence in each prediction. Subjects were almost equally confident in their self predictions (with a median value M of 84%) and in their predictions of others (M = 83%). However, they were notably more accurate in predicting their own behavior than predicting the behavior of others (M = 81% versus M = 68%, respectively). This made Griffin and Tversky claim that people "exhibited considerable overconfidence in predictions of others," while they were "relatively well–calibrated in predicting themselves."

There are also situations where predictions of one's own behavior may be underconfident. Griffin and Tversky asked a number of colleagues that were choosing between job offers if they could estimate the probability that they would choose one job over the other. The average confidence in the predicted choice was 66%, but in the end, 96% retained their predicted choice. It seems thus clear that even a small advantage for job X over job Y make people choose the former.

Griffin and Tversky also compared confidence and expertise when making predictions. In their experiment, the subjects where given randomly selected pairs of American states and told to choose the state that was higher on a particular attribute and to assess the probability of their answer being correct. The three attributes were the number of people in each state (Population), the high–school graduation rate in each state (Education), and the difference in voting rates between the states in the last presidential election (Voting). Griffin and Tversky expected people to be more knowledgeable and confident about Population than about Education and Voting. They also expected people to be more confident about Education than about Voting, because of typical stereotypes about one state being more "educated" than the other. Such stereotypes arise when a state has more famous universities and is associated with cultural events, even though the correlation between these cues and the high–school graduation rates is very low. Griffin and Tversky thus expected high accuracy and high confidence for Population, low accuracy and low confidence for Voting, and low accuracy and higher confidence for Education.

To test their hypothesis, Griffin and Tversky made 298 subjects each evaluate 15 pairs of states on one of the attributes. After evaluating all of the attributes, the subjects were told to estimate the number of attributes that were correctly guessed, and they were reminded that by chance alone the expected number would be 7.5. Table 3.2 shows the average confidence and average accuracy for the three attributes.

Griffin and Tversky claim the observed pattern to be consistent with their hypothesis. For Population, people had an accuracy of 68.2 and a confidence of 74.7. For Voting, the accuracy was 51.2 and the confidence 59.7, while the numbers were 49.8 for accuracy and 65.6 for confidence when it came to Education. It was thus a much greater

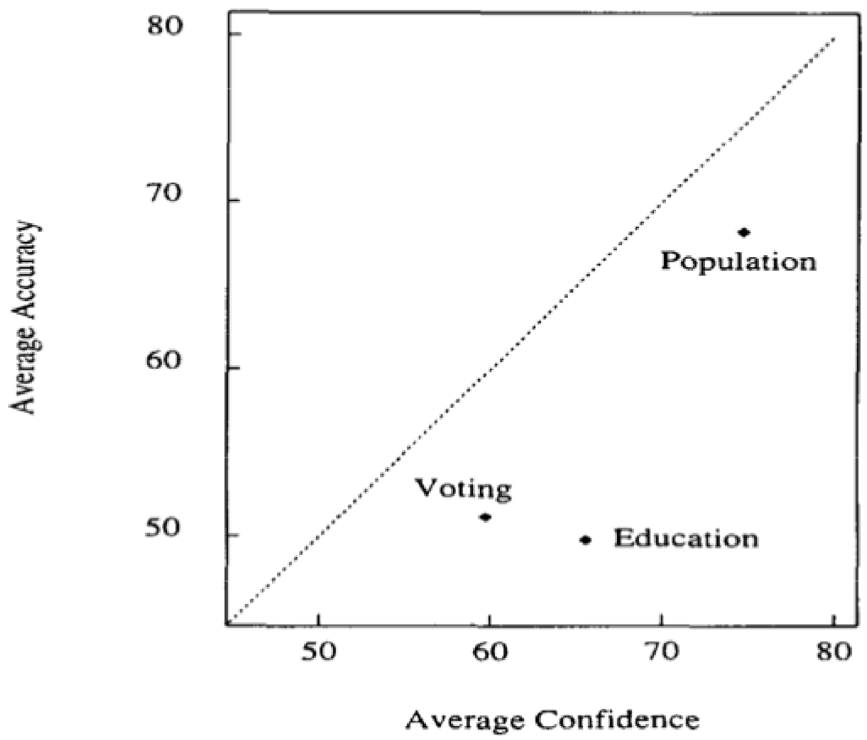


Figure 3.2: Excerpt form Griffin and Tversky: Average confidence and average accuracy for the three attributes [19].

overconfidence in judgements about Education than about Voting, even though the accuracies were close to identical. The overconfidences in Voting and Population were comparable.

According to Griffin and Tversky, their analysis shed light on the relation between overconfidence and expertise. They claim that "when predictability is reasonably high, experts are generally better calibrated than lay people," but when "predictability is very low, however, experts may be more prone to overconfidence than novices." Griffin and Tversky state that their analysis is consistent with cited studies of race odds-makers, expert bridge players, clinical psychologists, and stock market analysts.

Griffin and Tversky also look at the neglect of base rates. A striking example is found in the optimism that entrepreneurs express in their own ventures. Entrepreneurs are, on average, highly optimistic ("overconfident") about the success of their specific new ventures, even though they are reasonable realistic about the general rule of failure for ventures of that kind. Griffin and Tversky write that "the tendency to prefer an individual or 'inside' view rather than a statistical or 'outside' view represents one of the major departures of intuitive judgement from normative theory."

3.3.3 Tversky and Kahneman : Belief in the Law of Small Numbers

In 1971, Tversky and Kahneman looked into how people have erroneous intuitions about the laws of chance [48]. They claim that people have a perception that a randomly drawn sample from a population is highly representative, and that people believe that any two samples drawn form a population is more similar to another than sampling theory predicts.

Tversky and Kahneman deployed a questionnaire at meetings of the *Mathematical Psychology Group* and the *American Psychological Association*. The subjects were given several scenarios, including the following:

> Suppose you have run an experiment on 20 subjects, and have obtained a significant result which confirms your theory ($z = 2.23$, $p < .05$, two–tailed). You now have cause to run an additional group of 10 subjects. What do you think the probability is that the result will be significant, by a one–tailed test, separately for this group?

The median answer of the two groups was 0.85. However, careful statistical analysis shows that the most reasonable estimate is 0.48. Only 9 of 84 respondents gave answers between 0.40 and 0.60, and similar questions (about statistics and sampling) to the same groups gave similar results.

Tversky and Kahneman claim that their questionnaire "elicited considerable evidence for the prevalence of the belief in the law of small numbers," and that there is "practically no differences between the median responses of audiences at a mathematical psychological meeting and at a general session of the American Psychology Association convention (...) Apparently, acquaintance with formal logic and with probability theory does not extinguish erroneous intuitions."

3.4 Summary

This chapter has shown some of the most relevant related work. Cai and Card (section 3.1) summarized the subjects of journal and conference papers in software engineering, and found that the most popular topics are "testing and debugging", "software / program verification", "management", and "design tools and techniques".

Osterweil et al. claim that research in software engineering has a substantial impact on practice, but that there is a widespread perception that this is not not the case (section 3.2). This notion could be explained by the findings of Redwine and Riddle, who found that the average time for a key idea to move from research to practical applicability was 17 years.

The works cited by Tversky, Kahneman, Griffin, and Frederick in sections 3.3.1, 3.3.2 and 3.3.3 show that experts and intuition often are unreliable. People tend to believe in a "law of small numbers", and acquaintance with formal logic and with probability theory does not extinguish erroneous intuitions — something which is unfavorable to experts, who one could expect to know better. People are also violating basic logical rules when relying on representativeness when providing probability judgments, something that was illustrated by the famous "Linda" problem. Also, people tend to be overconfident when making predictions, and neglect known base rates. These issues have a strong impact on SRQ 2, as they show that experts' advice and intuition — the two strong competitors to research and analysis when making decisions — have some significant, negative properties.

Chapter 4

Methodology

This chapter describes why we chose a survey, and how we conducted, deployed and administered it.

Section 4.1 gives a very short overview of some groups of research methods in empirical software engineering. Section 4.1.1 describes in general the method of our choice; the personal opinion survey.

Our questionnaire specification is found in section 4.2. This section includes all of our questions (in an annotated form), and rationales for each question. Section 4.2.3 describes our evaluation process, where we in an iterative manner carried out two rounds of pilot testing. There were unfortunately some errors that we did not catch in this process, these are described in section 4.2.4.

Section 4.3 describes how we administered the survey. We used a framework called *SurveyMonkey* for collecting data, and deployed the survey in one users' group and two companies. Section 4.3.3 briefly discusses the quality of the data we collected.

4.1 An overview of empirical software engineering methods

Empirical software engineering utilizes a wide array of research methods, each with its own strengths and weaknesses. The different research methods can be grouped in various ways. For example, Singer at al. divide data collection methods for field studies into three main groups: *direct techniques*, *indirect techniques*, and *independent techniques* [43]. The direct techniques (which are the most interesting to us) are further divided into *inquisitive* techniques (e.g. brainstorming, interviews, and questionnaires), and *observational* techniques (e.g. work diaries, think aloud sessions, and participant observation). According to Singer et al., "inquisitive techniques allow the experimenter to obtain a general understanding of the software engineering process", and they also claim that "such techniques are probably the only way to gauge how enjoyable or motivating certain tools are". Observational techniques, on the other hand, "provide a real-time portrayal of the studied phenomena", but also make it "more difficult to analyze the data, both because it is dense and because it requires considerable knowledge to interpret correctly". Observational techniques may also lead to a change of process simply by observing it, an effect known as the Hawthorne effect.

Both inquisitive and observational techniques could be suitable for answering our RQ and SRQs. The disadvantages with observational techniques (Hawthorne, possible difficulties with analyzing data) have made us choose inquisitive techniques for this master thesis. Among the inquisitive techniques, many techniques are suitable for our research. Table 4.1 shows the main areas of application for the five most common inquisitive techniques. As one can see, interviews and questionnaires are the most useful techniques for finding opinions, as we are going to do in our research question.

Technique	**Usage**
Brainstorming and focus groups	Understanding ideas and general background about the process and product, general opinions
Interviews and questionnaires	Finding general information (including opinions about process, product, personal knowledge etc.)
Conceptual modeling	Finding mental models of product or process
Work diaries	Finding time spent or frequency of tasks, pattern of activities, some goals and rationale
Think–aloud sessions	Finding mental models, goals, rationale and patterns of activities

Table 4.1: Excerpt from the summary of data collection techniques by Singer et al. [43]

We were initially considering using a combination of interviews and questionnaires in this master thesis, and were planning on using a structured interview technique called *repertory grids*. However, logistical considerations made us focus fully on the questionnaire approach, as the latter made it significantly easier to reach a good excerpt of the population of software practitioners.

4.1.1 Surveys : an overview

As defined by Fowler, the purpose of a survey is to gather quantitative or numerical descriptions about some aspects of a study population [14, p. 1 ff.]. This is achieved by asking people questions, and the answers constitute the data to be analyzed.

Data for a survey can be collected in several ways. Traditionally, surveys involved face-to-face sessions, a practice which has some weaknesses. Most important of the weaknesses is the possibility for the interviewer to influence the respondent so that he or she could change the answers, introducing important biases. This has been known for a long time, for example from an experiment by Stanton et al. who concluded that "the bias of the interviewer exerts some determining effect upon the outcome of the interview even when the interviewer is experienced, the direction of the bias known to him, and the material has no personal or emotional connection" [46]. (The experiment was later replicated by Friedman, who could not confirm the findings of Stanton et al. [16]).

In the 1970s, as telephone ownership increased, telephone interviewing became a major mode of data collection [14, p. 7]. Telephone interviews eliminate some of the interview's weaknesses, such as for example the effect of the body language of the interviewer. Later, the Internet proved to be an even more suitable mode of survey data collection. The use of the Internet for surveys could among other things extend access to participants, defuse the embarrassment that some subjects may feel when asked sensitive questions by the interviewer, and lead to easier handling of data [32, pp. 17–25]. There are of course also some disadvantages to Internet surveys, such as that Internet access is not evenly distributed among socioeconomic groups, something which may lead to a selection bias. The latter is nonetheless not a problem in our survey, as one can reasonably expect that all or nearly all software practitioners have access to the Internet.

A survey cannot reasonably cover all members of a population. There are some exceptions to this (such as the U.S. decennial census), but one does generally choose only a small, representative part of population to participate in a survey — a practice known as *sampling*. In our survey, we wanted to describe the population of software practitioners — this should be reflected in the sample, which ideally should include all kinds of practitioners from all kind of software companies.

Fowler describes several different sampling techniques [14, pp. 14–28]. Among these are *simple random sampling*, that approximates drawing a sample out of a hat: members of a population are selected one at a time independent of one another and without replacement; *systematic sampling*, a less laborious variant of simple random sampling; and *stratified sampling*, where the researcher deliberately chooses subjects so that the sample will have exactly the same proportions as the whole population.

It would have been extremely hard for us to get a sample that was randomly distributed among the population. Our study thus used a more informal, *ad hoc*-approach where availability was the most important consideration. See section 4.3.2 for the details on how we found subjects for our survey.

Best practices

Fowler lists five basic characteristics of the question of a survey that are necessary in order to get a good measurement process [13, pp. 4–5]:

1. Questions need to be consistently understood
2. Questions need to be consistently administered or communicated to respondents
3. What constitutes an adequate answer should be consistently communicated
4. Unless measuring knowledge is the goal of the question, all respondents should have access to the information needed to answer the question accurately
5. Respondents must be willing to provide the answers called for in the question

Fowler mentions three main steps that can be taken to assess to which extent the questions meet these standards: (1) focus groups discussions, (2) intensive individual interviews, and (3) field pretests under realistic conditions [13, p. 104]. For our purpose, we chose a pilot study in two iterations, an approach that is closest to (3). The pilot has been described in detail in section 4.2.3. We believe that the pilot will sufficiently take care of characteristics 1–4, while characteristic 5 probably is of minor concern, as none of our questions are likely to relate to highly sensitive matters.

Survey documentation

Kitchenham and Pfleeger describe a document called a *questionnaire specification*. This document is best written before the questionnaire is administered, and should include [30]:

1. The objective(s) of the study
2. A description [of] the rationale for each question
3. The rationale for any questions adopted or adapted from other sources, with appropriate citations
4. A description of the evaluation process

When the questionnaire is administered, the documentation should be updated to record information about:

1. Who the respondents were
2. How it was administered
3. How the follow-up procedure was conducted
4. How complete questionnaires were processed

Preparing this documentation ensures that the researchers do not forget the details of instrument creation and administration.

We have produced survey documentation as suggested by Kitchenham and Pfleeger. The documentation can be found in sections 4.2 and 4.3.

4.2 Questionnaire specification

The following subsections consist of our initial specification of the survey, as recommended by Kitchenham and Pfleeger and summarized in section 4.1.1. The specification is reproduced verbatim, with a few exceptions for the sake of readability. We omitted a section for the rationales for questions adopted or adapted from other sources, as there were no such questions.

The original questionnaire (without annotations) can be found in appendix A.

4.2.1 Objectives

The objective of our survey was to answer the research question that is found in section 1.2, hereunder SRQ1– SRQ4.

4.2.2 Questions and rationales

The following sections consist of the questions for the questionnaire, with annotations that describe our rationales and motivations for including them. In this specification, we have omitted the part of the questionnaire that asks about demographical data.

Question 1

> 1. You are considering a new technique or technology in IT development. You have only superficial prior knowledge about this technique or technology.
>
> On a scale where 1 is "not at all" and 7 is "very much", how important for you is advice from the following persons and entities when you are making your decision?

This question was aimed at answering SRQ2 ("do software practitioners value experts' advice more than they value research"). We deliberately avoided to use a specific example of a technique or technology, in order to make the question more universal and suitable for all kinds of software practitioners.

It could be that the respondents concretize the question by substituting "a [unspecific] new technique or technology" with a particular technology they are used to or a technology choice they have made recently. This would be unfortunate, as that choice could have been influenced by other variables, outside of our control. As always, researchers have to compromise when designing surveys, and we chose to use a unspecified technique or technology for the sake of generalization.

> A colleague or friend with experience with this technique

In everyday life, it is natural to take the opinions of one's friends and colleagues into consideration. It is thus interesting to see if this also applies when making professional decisions.

We implicitly meant "a friend with experience from the software industry" when we wrote "friend". During our pilot study, all subjects shared this implicit condition.

> An external expert / guru

It was obvious to include this option, as experts' are explicitly mentioned in SRQ2.

To define an "expert", we have used some definitions from Meyer and Booker [34, p. 3]:

> Expert judgement is data given by an expert in response to a technical problem. An expert is a person who has background in the subject area and is recognized by his or her peers or those conducting the study as qualified to answer questions. Questions are usually posed to experts because they can not be answered by other means.

Following the definitions of Meyer and Booker, an "expert" or "guru" must in our context be understood as someone who the respondent recognizes as qualified to answer a question regarding a technical problem, when the question is too difficult for the respondent to fully or satisfyingly answer himself.

> Independent researchers (MIT, The Norwegian University of Science and Technology)

and

> Industrial researchers (IBM, Sun) and commercial research companies (The Standish Group)

We decided to split "research" as mentioned in SRQ2 into two options, as we consider research made by independent researchers to be vastly different in nature from research made by commercial entities.

We assumed that all Norwegian software practitioners know the Norwegian University of Science and Technology, perhaps the premier college for the technological sciences in Norway. MIT (the Massachusetts Institute of Technology) should also be well known. We were also considering adding a non-university research organization, namely SINTEF. SINTEF is probably well known among Norwegian practitioners in the ICT industry. The organization is however more renown for non–ICT disciplines (such as construction, materials and chemistry, economics, petroleum, and marine technology), so we ultimately decided against mentioning them, as this survey is about ICT.

It was harder to find suitable, well known examples of commercial research entities, apart from major companies that have large R & D divisions (IBM, Sun). We chose The Standish Group because of their highly cited CHAOS report, but we do not really expect many software practitioners to have heard about them.

> Information from the vendor or supplier (success stories, product demonstrations)

Success stories and product demonstrations are also possible sources of knowledge for policy makers.

> Other (please specify)

There are many additional sources of information, so this option is useful here.

We decided against requiring an answer to question 1. Unanswered questions can be interpreted as “do not know” type answers, even though we did not instruct the respondents in this matter.

Question 2

> 2. When looking for knowledge about new development techniques and technologies, to what extent do you consult the following sources?

This question is directly derived from SRQ1 (“How do software practitioners learn about new techniques and technologies?”).

> Experts/gurus, colleagues, and friends in the IT industry

The same rationale applies here as in question 1.

> Industrial conferences (JavaZone etc.)

JavaZone is a Norwegian conference for Java developers. We believe it to be perhaps the most widely known Norwegian industrial ICT conference.

We were initially concerned that the respondents did not differentiate between industrial conferences and scientific conferences, but our pilot showed that the respondents had a good grasp of the difference.

Popular scientific journals and magazines

We were originally naming some examples of popular scientific journals and magazines. We used *IEEE Software* as our prime example. This is a bimonthly magazine that aims to deliver "reliable, useful, leading-edge software development information to keep engineers and managers abreast of rapid technology change" [21]. We later decided against using *IEEE Software* as an example, as we do not really expect a majority of practitioners to have heard about it.

During the first iteration of our pilot study, we asked one of the participants if she could give an example of a popular scientific magazine. Her response was *Teknisk Ukeblad*, a Norwegian magazine which we would put under the "Websites and IT newspapers" option (see below). We thus assume that there is some disagreement in the population about what a "popular scientific magazine" is, but an unambiguous definition of this concept is not critical for our survey.

Scientific conferences

We chose not to provide any examples here. We did not expect many of the respondents to learn about new techniques and technologies at scientific conferences, but we expected the practitioners who actually attend such conferences to know what a scientific conference is.

Scientific journals (IEEE Transactions on Software Engineering etc.)

Scientific journals could be a source of knowledge for professionals. We included an example to help the subjects differentiate them from non–scientific journals.

Web forums, mailing lists, blogs and other social media

The World Wide Web is clearly a source of information for IT professionals. We also assumed that many gain information from social media, such as Twitter.

We did not provide any examples of social media, as we expected everybody to understand what this is. This item was added after the pilot, so we could not get feedback on our assumption.

Websites and IT newspapers (Slashdot, Computerworld)

Computerworld is a Norwegian weekly newspaper, which claims to be Norway's "largest and oldest ICT Newspaper". We did not look for external citations that could verify this claim, but it was probably safe to assume that this newspaper is well known among software practitioners and thus a safe choice as an example.

Slashdot is a technology oriented news website. It incorporates elements from social media, and its "comments" sections often include hundreds of user submitted comments. We would however use it as an example of a news site instead of lumping it together with the "web forums, mailing lists, blogs and other social media" option.

Other (please specify)

The same applies here as for question 1.

We did not require an answer to question 2.

Question 3

> 3. It has been decided that your organization needs to radically renew its software development process model. You are responsible for evaluating the alternatives, and there are two obvious alternatives, X and Y. You have assessed the available literature, and while there are pros and cons to both alternatives, your analysis of the literature shows that alternative X is probably the best choice for your organization.
>
> Nevertheless, you have a strong feeling (gut feeling, "magefølelse") that alternative Y is best suited for your organization, even though you find it hard to explain why.
>
> On a scale where 1 is "gut feeling only" and 7 is "analysis only" how much weight would you give to your analytical findings versus your gut feeling?

This question is related to SRQ 2. SRQ 2 is concerned with how practitioners value experts' advice compared to how they value research, and it is clearly interesting to compare and contrast this to how they put weight to gut feeling.

The "Linda" problem and the trustworthiness of intuition that we discussed in chapter 3, section 3.3 are highly relevant to this question. "Linda" showed that intuition can violate basic rules of logics, something which a careful analysis of a problem should not do.

We deliberately chose "X" and "Y" to describe the alternatives instead of descriptions as "A" and "B" or "1" and "2", as "X" and "Y" are neutral terms and do not imply that one alternative is better than the other.

When we deployed the survey, we became aware that this question had a potentially serious typographical error that we did not catch during the pilot. See section 4.2.4 for details.

Question 4

> 4. Your organization needs a new testing framework. You have found reliable research claiming that using framework F will most of the time find more bugs and increase productivity.

> You have also consulted an expert whom you trust from previous experience. He claims that framework F is worthless, and strongly recommends using framework G instead.
>
> On a scale from 1 (“research only”) to 7 (“expert advice only”), how would you give weight to the two recommendations?

This question concerns SRQ 2.

The question was modified after the evaluation process (see section 4.2.3). We originally intended to use TDD (Test Driven Development) as an example, and the original wording was “Your organization is considering using test driven development (TDD) in a large scale. You have found reliable research showing that TDD most of the time find more bugs and increase productivity (. . .) ”.

We decided against using a specific example of a technique after the second iteration of the pilot survey. Both subjects in this evaluation round had strong opinions about TDD, and both claimed that their experience was that TDD indeed increased productivity and found more bugs. We thus changed the wording to a neutral “testing framework” to avoid introducing a possible bias.

Note that the original formulation was “Your organization *is considering*” making a switch. Such a formulation is unfavorable, as one can easily imagine a manager to be conservative and prefer a familiar yet suboptimal solution in favor of something new and untested. We consequently changed the formulation so that the decision to switch was already made.

Like for question 3, we chose neutral terms like “F” and “G” to describe the frameworks instead of “A” and “B” or “1” and “2”.

Question 5

> 5. On an scale from 1 (“not at all”) to 7 (“very much”), how much do you keep up with research in software engineering?

This question is related to all of the SRQs. How much practitioners keep up with research in software engineering can certainly influence how they make decisions, and we suspect that practitioners who closely keep up with research are more analytical in their decisions than practitioners who do not keep up with research.

The question initially had an grammatical error, see section 4.2.4 for details.

Question 6

> 6. On an scale from 1 (“nothing”) to 7 (“very much”), how much confidence do you have in research in software engineering?

This question arises out of SRQ 3, but is also closely related to the main RQ ("how do software practitioners value research when making decisions"). It is reasonable to hypothesize that practitioners who have more confidence in research value research more when making decisions than practitioners who do not have confidence in research do.

Question 7

7. Do you apply results from research in software engineering in your work?

- Yes
- No
- Don't know

This question is also related to the main RQ. It is likely that practitioners who have confidence in research more often apply results from research in their work, and the opposite is also likely to be true.

Question 8

8. Do one or more of these barriers prevent you from applying research in your work? (choose all that apply, choose none if no alternative applies)

- My organization does not encourage the use of research
- Too busy meeting immediate goals or deadlines
- Research results are hard to find
- Lack of personal time
- Research is not relevant to my needs
- Attitudes of colleagues
- Attitudes or expectations of customers
- Other (please specify)

Question 8 is directly derived from SRQ 3. The options were adapted from the barriers used by McColl et al. when they determined medical practitioners attitudes towards evidence–based medicine (see section 2.4), as it can be interesting to compare software practitioners' and medical practitioners' reasons for not applying research.

Question 9

> 9. Name up to three topics that you feel software engineering researchers should focus on more.

This question is directly taken from SRQ 4.

We initially considered providing a list of topics for the participants to rank, using the SE topics of ACM's Computing Classification System [2]. We lated changed our mind, instead asking the respondents to express themselves more freely, not using predefined categories. This made the questionnaire considerably shorter, giving more space to other, more relevant questions. It is also unlikely that all respondents have opinions about the state of all areas of SE research, and presenting all topics could have given "don't know" or random answers *en masse*.

Question 10

> 10. Which software engineering activity or job task is your main daily focus at present?

This question acts as a "check and balance" to Question 9, as it is likely that respondents often feel that it is their daily activities researchers should focus on.

4.2.3 The evaluation process

We conducted two iterations of field pretesting ("pilot testing") of the survey. The intention of such a process is to discover practical problems, such as typographic errors, faulty instructions, and inadequate arrangements for recording answers [13, p. 115].

A pretest is done when the survey instrument is in a near final form. Fowler recommends conducting 15 to 35 interviews with persons similar to those who will be respondents in the planned survey, using similar data collection procedures [13, p. 115].

15 to 35 interviews would be way too many for our limited study, as we expected somewhere between 50 and 100 respondents. We decided upon using 4–6 interviews for the pretest, split into two rounds. This iterative approach lessens the risk of introducing new errors and ambiguities.

Round one

For the first round, we used three college students as subjects. They were in their twenties, and had no industry experience. All were enrolled into a computer science program at the University of Oslo, and were senior students in the final year of their bachelor's degree. They were told that the process would take approximately 15 minutes, and were promised a beer as a reward for participating.

We did the three pilots sequentially in a quiet, dedicated room. The subjects were told that the survey was about how software practitioners made decisions, but we specifically avoided telling that it was about how they valued research. We asked the subjects to read the questions aloud, and to "think aloud" when answering. We also asked additional questions, such as "what is a 'software development process model'?". The goal of asking the additional questions was to make sure that the subjects had a common understanding of the most important terms used in the survey.

We learnt many lessons during this round of pilot testing. First of all, all three subjects failed to mention even a single topic that software engineering researchers should focus more on (Question 9). When we asked them why, they answered "dont't know". We then asked if they could mention some random topics in software engineering. All were hesitating, but could after some consideration mention a couple of topics each ("databases", "programming", "project management", "modeling", "problem analysis", and "testing" were mentioned, all of which are valid SE topics). Because of the subjects' inability (or reluctance) to answer question 9, we were considering removing it. However, we chose to keep it for the second iteration, as practitioners are more likely to be able to name SE topics than college students are.

All subjects shared a common understanding of what an "external expert or guru" is, and described it as someone who was renown within the particular field. They did not specify if this status was among practitioners or scientists (or both). All subjects were also asking themselves if the "an experienced colleague or friend" option referred only to colleagues and friends working in ICT.

It was also clear that the test subjects did not know what a scientific journal is. One of the subjects stated that popular scientific journals "are *about* stuff", while scientific journals concern "how to *use* stuff". Their confusion was was hardly surprising, as the subjects were undergraduates. All subjects answered "1 – not at all" for "scientific journals" in question 2, and we decided that the lack of understanding of what a scientific journal is isn't a problem as long as the subjects answer "not at all". If they do not know what scientific journals are, they are extremely unlikely to use them to learn about new development techniques and technologies.

For question four, two of the three test subjects were unsure about what TDD is. When we asked them to make a guess, one said "testing during development", while the other said "using automated tests instead of making users test the system." We presumed that such misunderstandings were significantly rarer among practitioners, and decided against switching to another example at this stage.

For question five and six, all test subjects had an adequate understanding of what "software engineering" is. The first subject left question six blank and said he did not know, the second subject said he did not know and gave it a four, and the third subject said he would give it a five or a six. At the time, question six instructed the subjects to "choose one, leave blank if you do not know". Subject two thus misunderstood the instructions, but we decided to await the second iteration of the pilot before possibly clarifying.

The test subjects did not have any particular comments about question eight, but two of them noted that “it is too much of a hassle” was the obvious choice. This option was removed from the final version of the survey, but we kept it for the second round of pilot testing.

Round two

For the second round of pilot testing, we found two volunteers in a mid-sized Norwegian company that make ERP systems. Test subject one had a variety of responsibilities, including development, QA, and deployment, while test subject two was solely doing programming.

As in the first round, we did the second round of pretesting as “think aloud” sessions, one subject at a time. We did not offer any reward for participating this time.

Both test subjects interpreted an “expert / guru” as somebody independent, without ties to a vendor or supplier. One of the subjects said that one had to have more than ten years of industry experience to be considered an expert. He also asked if “an experienced colleague or friend” (in Question 1) meant someone with experience with this particular technique or technology, or someone with more general experience. Similar comments were made in round one, and we therefore changed the wording accordingly to “a colleague or friend with experience with this technique”.

One of the test subjects was initially confused about the “industrial conferences” source of knowledge (Question 2). He first gave it a “1 – not at all”, but when he saw the next row (“scientific conferences”), he changed his response for “industrial conferences” to a five. He then said that he often learnt about new technologies at Microsoft workshops, and that “industrial conferences” was the most fitting category for such events.

Test subject two remarked that a “gut feeling” could be both rational and irrational, and asked if this perhaps should be defined clearer in Question 3. We agreed to some degree, but nevertheless chose to leave the question in its original form. Test subject one found the question too long.

Both test subjects said that they do not keep up with research in software engineering, and that they thus felt that “four” was the natural option, as they did not know. This made us change the “choose one, leave blank if you do not know” instruction into “choose one”, with an explicit “don’t know” option to the far right. After some discussions with other researchers, we later chose to revert this change. The reason was that some participants could mistake the “don’t know” option with “7 – very much”, which normally is the option to the far right. We thus decided that standardization of the questions is more important than making the “don’t know” option clearer.

For question eight, both test subjects said that “it is too much of a hassle” was the obvious choice, and that it perhaps was too broad. We consequently removed this option, so that the subjects would be forced to give more specific reasons.

For questions nine and ten, the test subjects claimed that they had a hard time understanding what we meant by “topic” and “job task”. Test subject two simply answered “software engineering” as his job task, something which was way more general than what we wished for. Test subject one was more to the point, and answered "installation / deploy". We were considering introducing some examples of what we considered software engineering topics ("testing", "requirement elicitation" etc.) to give the subject a gentle push in the right direction, but we later changed our minds and removed the examples. This was because we were afraid that the subjects simply would choose among the provided examples instead of giving more precise answers.

4.2.4 Errors uncaught during the evaluation process

After the second iteration of the pilot, we decided to change "two alternatives, A and B" into "two alternatives, X and Y" in Question 3. This was because "alternative A" may sound superior to "alternative B". However, when making the change, we made a copy and paste error. This caused the question to initially have the following erroneous formulation (the error is emphasized in bold):

> 3. It has been decided that your organization needs to radically renew its software development process model. You are responsible for evaluating the alternatives, and there are two obvious alternatives, X and Y. You have assessed the available literature, and while there are pros and cons to both alternatives, your analysis of the literature shows that alternative X is probably the best choice for your organization.
>
> **You have assessed the literature you have available, and while there are pros and cons of both alternatives, your analysis shows that alternative A probably is the best choice for your organization.**
>
> Nevertheless, you have a strong feeling (gut feeling, "magefølelse") that alternative Y is best suited for your organization, even though you find it hard to explain why.
>
> On a scale where 1 is "gut feeling only" and 7 is "analysis only" how much weight would you give to your analytical findings versus your gut feeling?

The first 20+ responses were collected using paper copies of the questionnaire. We were available for questions, clarifications and feedback at this time, but received no comments on the error. It was not until we started using a web version of the questionnaire that we were told about the error. At that time, we had collected some 40 responses. We then fixed the phrasing, so that it was in line with our intentions.

We also found another error in question five, which said "(...) how much do keep up with research (...)" instead of the intended "(...) how much do **you** keep up with research (...)". We consider this error minor, at least compared to the error in question three. This error was fixed after we had received approximately 100 responses.

4.3 Questionnaire administration

4.3.1 The survey framework: SurveyMonkey

We had several options regarding which web framework we could use for collecting data. We considered using Simula research Laboratory's *SESE* ("Simula Experiment Support Environment"), but in the end decided upon using *SurveyMonkey* (`http://www.surveymonkey.com`), a survey framework which has been recommended by several independent reviewers. For example, Gordon describes *SurveyMonkey* as "an excellent survey and evaluation tool for online learning environments and for research in the field of online learning environments" and claims that it is "easy to configure, has a rich array of options, and is easy to work with" [18]. It is also easy finding numerous published, peer–reviewed studies that use it. Some, such as Abel et al. have noted that *SurveyMonkey* "is not primarily targeted for academic research purposes", but nevertheless claim that "[t]he use use of *Survey Monkey* reinforces principles of effective surveys as outlined by Dillman (2000), such as allowing questions in conventional formats" [1].

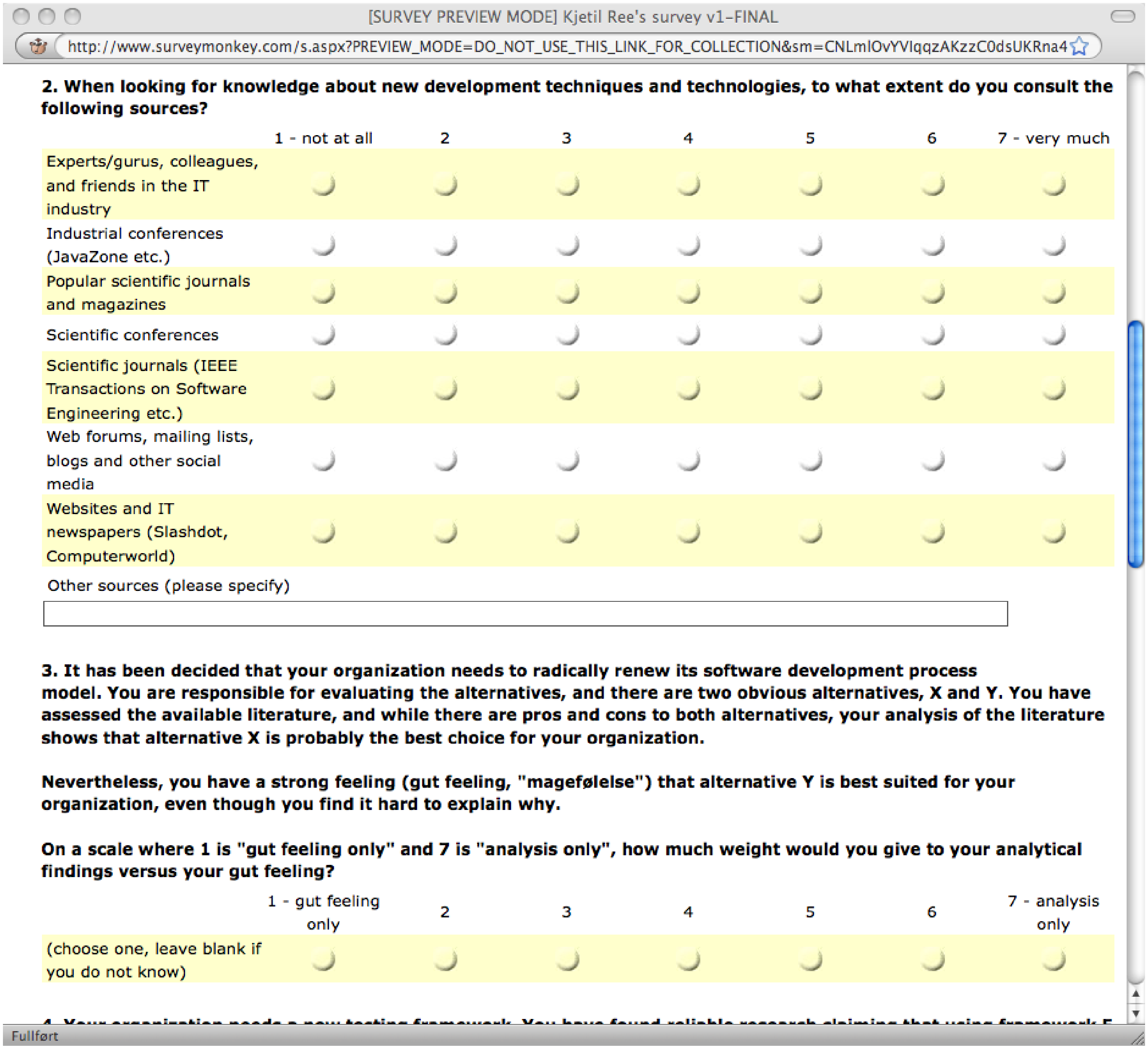
SurveyMonkey supports over twenty types of questions, including multiple choice, rating scales and open–ended text. This was more than sufficient to meet our needs. All questions could have their layouts customized, and professionally designed survey templates were also available.

We designed the survey so that it had three distinct parts. Page one was introductory, explaining that this was a survey from the software engineering department at Simula Research Laboratory, and who it was aimed at. It also prompted for the participants' email addresses, as one of them would win an Apple iPod. Stating one's email address was however totally voluntary, and the participants could choose to be anonymous if they wanted to.

Page two asked for some basic demographical information, such as age, location, position, industry experience, and education. It was mandatory to answer all of these questions.

The actual survey (as described in section 4.2.2) came on page three. Figure 4.1 shows how the third page of the survey appears to the respondents. None of the questions here were mandatory. Figure 4.2 shows how the questions were designed in *SurveyMonkey*.

SurveyMonkey allows several *collectors*. A collector is basically a link that can be sent to an audience. We created several such collectors, one for each organization that we deployed the survey in (see section 4.3.2 for more about the deployment). Using disjunct collectors allowed us to keep the data collected from each organization separate. Collectors can be of several types; we chose the type called a "web link". Such links are *not* unique for each participant, and are sent to the participants in an email or posted on at web site. This allows the survey to be completely anonymous if the respondent wishes to, with the one exception that the IP address associated with each respondent is stored.


[SURVEY PREVIEW MODE] Kjetil Ree's survey v1-FINAL
http://www.surveymonkey.com/s.aspx?PREVIEW_MODE=DO_NOT_USE_THIS_LINK_FOR_COLLECTION&sm=CNLmlOvYVlqqzAKzzC0dsUKRna4
Fullført

2. When looking for knowledge about new development techniques and technologies, to what extent do you consult the following sources?

	1 - not at all	2	3	4	5	6	7 - very much
Experts/gurus, colleagues, and friends in the IT industry	○	○	○	○	○	○	○
Industrial conferences (JavaZone etc.)	○	○	○	○	○	○	○
Popular scientific journals and magazines	○	○	○	○	○	○	○
Scientific conferences	○	○	○	○	○	○	○
Scientific journals (IEEE Transactions on Software Engineering etc.)	○	○	○	○	○	○	○
Web forums, mailing lists, blogs and other social media	○	○	○	○	○	○	○
Websites and IT newspapers (Slashdot, Computerworld)	○	○	○	○	○	○	○

Other sources (please specify)

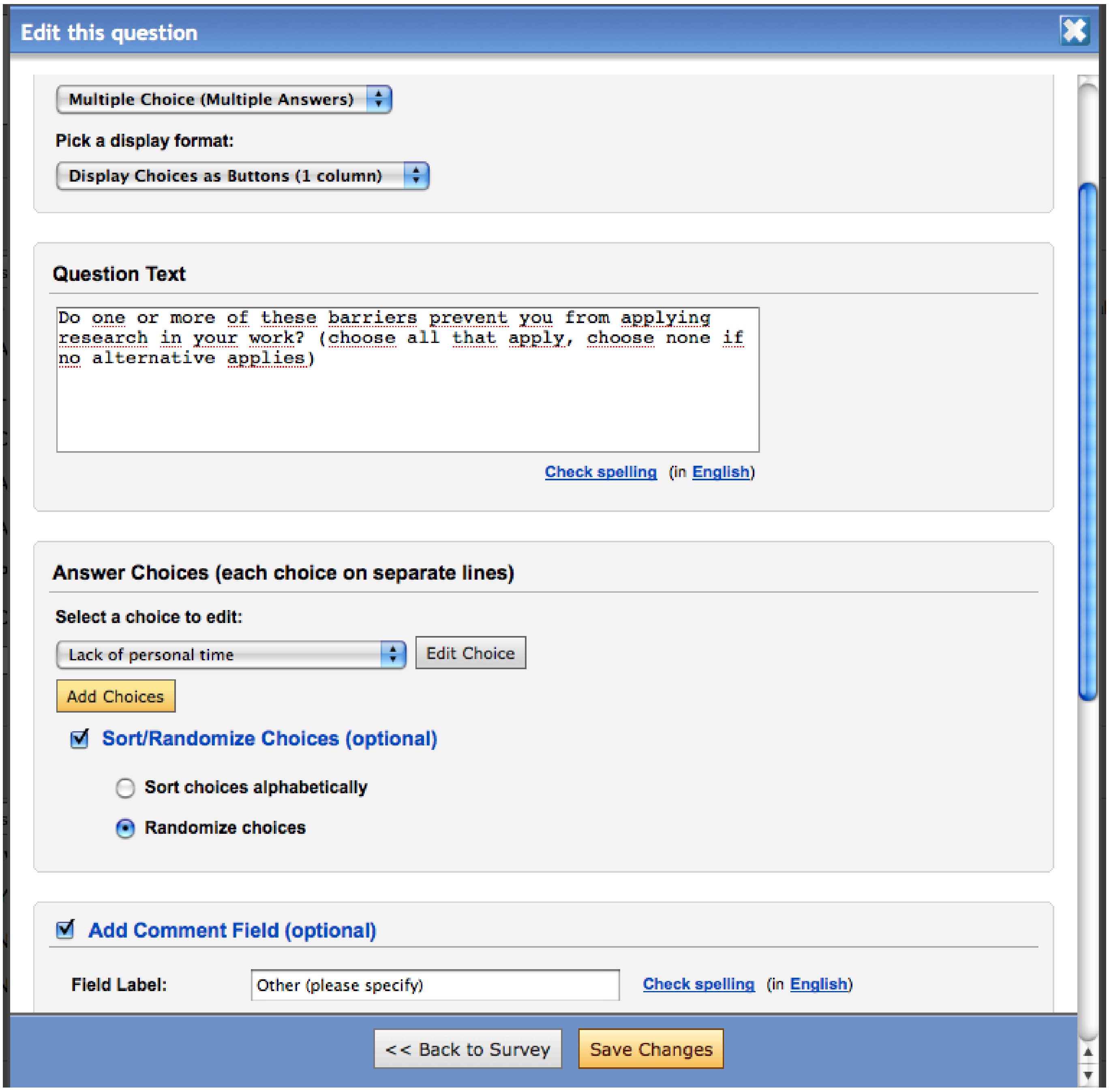
3. It has been decided that your organization needs to radically renew its software development process model. You are responsible for evaluating the alternatives, and there are two obvious alternatives, X and Y. You have assessed the available literature, and while there are pros and cons to both alternatives, your analysis of the literature shows that alternative X is probably the best choice for your organization.

Nevertheless, you have a strong feeling (gut feeling, "magefølelse") that alternative Y is best suited for your organization, even though you find it hard to explain why.

On a scale where 1 is "gut feeling only" and 7 is "analysis only", how much weight would you give to your analytical findings versus your gut feeling?

	1 - gut feeling only	2	3	4	5	6	7 - analysis only
(choose one, leave blank if you do not know)	○	○	○	○	○	○	○

Figure 4.1: Screenshot from *SurveyMonkey* : how the survey appeared to the respondents.


Edit this question

Multiple Choice (Multiple Answers)

Pick a display format: Display Choices as Buttons (1 column)

Question Text

Do one or more of these barriers prevent you from applying research in your work? (choose all that apply, choose none if no alternative applies)

Answer Choices (each choice on separate lines)

Select a choice to edit: Lack of personal time

[✓] Sort/Randomize Choices (optional)

Sort choices alphabetically [✓ Randomize choices]

[✓] Add Comment Field (optional)

Field Label: Other (please specify)

Figure 4.2: Screenshot from *SurveyMonkey* : designing the questions.

We set the collectors so that only one response was allowed per computer. The *SurveyMonkey* manual does not mention how the system enforces this; and we can easily think of several possibilities, where HTTP cookies and some sort of IP address filtering are the two most obvious. Some simple tests make us reasonably sure that HTTP cookies are used to enforce this restriction. This makes it trivial for respondents to participate several times if they so wish, but does not exclude persons who sit behind routers that use network masquerading (NAT) from participating fully.

The participants could not go back and update the responses after the survey had finished. They could however go back to previous pages and edit their responses before they finished the survey. The participants were redirected to a generic *SurveyMonkey* "thank you" page after they had finished.

SurveyMonkey makes generating paper copies of the survey easy, as it can convert the surveys into specially styled PDF versions. We gathered most of our data using the web collectors, but also used paper copies for some 20 respondents (see section 4.3.2 for details). The data collected using the paper copies was later punched into the system by hand.

SurveyMonkey allows the responses to be downloaded in a spreadsheet format. This makes it easy to migrate the data into the statistical software of our choice, namely JMP. See chapter 5 for more on how we analyzed the data.

4.3.2 Deployment

We were initially working on deploying our survey in a major software development company with thousands of employees, and offices throughout Europe ("Company X"). We were told that Company X was planning to deploy the survey to QA managers and developers in many countries, something which influenced some of the design choices we made. For example, we translated the survey from Norwegian into English after the pilots to accommodate non–norwegian subjects, even though this move reduced the value of the pilot process (but we nevertheless did our best to retain the enhancements the process created). We also included "QA manager" in the "In which position are you currently working" demographical question, and other similar adaptations.

All of our initial contacts in this company were positive, but the company unfortunately withdrew after our survey had been discussed at the executive level. This unfortunate withdrawal put us in a very difficult position, as we thus needed to find new subjects, and time was short before the general staff vacation started in June. We nevertheless do not hold a grudge against Company X, as we are fully aware that these things happen all the time when doing research.

We were able to find new respondents at *javaBin*, a Norwegian Java users' group. A users' group is a private, non-profit club where the members come together regularly to share knowledge on a particular technology or set of technologies, in this case Java and related technologies. We were on a short notice allowed to attend a member event, where we distributed paper copies of our survey. A link to the online version of our survey was also placed prominently on javaBin's web site. Nearly all members of javaBin (and thus the majority of visitors to their web site) are software professionals, and we urged the members not to participate if they were not professionals.

The short notice before the member event made it impossible to undo the changes we had done to the survey when preparing it for deployment in Company X. For example, some of the options were somewhat awkward — there is not likely to be many QA managers in javaBin — but this was of minor concern. Before attending the javaBin event, we added the possibility to win an iPod to the survey to introduce a stronger incentive to participate.

At javaBin, we also got in touch with some practitioners who wished to introduce the survey in their respective organizations. We pursued these opportunities, and thus found two more software development companies that were willing to deploy our survey in their organizations. The first company ("Company Y") is a Scandinavian consultancy company specializing in object oriented systems development, while the second company ("Company Z") is a Scandinavian company providing services in consulting, development services, training, support, and application management for a wide range of products. The survey was deployed in these two companies in a "semi-official" manner — that is, the deployments were authorized by the respective managements, but all the practicalities were carried out by our initial contacts. These contacts both sent appeals for participants to their respective companies' internal mailing lists.

In total, we got 145 respondents who started the survey, of which 113 completed it (77.9%). 82 came from javaBin, 23 came from Company Y, and 40 from Company Z. We closed the collectors three weeks after the javaBin event, and two weeks after we sent out the invitations to companies Y and Z. At that time, there had been three days since the last response.

This is an example of a *non–probabilistic sampling method*, as we chose respondents because they were easily accessible. The main problem with such *convenience sampling* is that the people who are available and willing to take part may differ in important ways from those who are not willing to participate [30, p. 86]. We nevertheless believe that the sample is good enough, as we can not see that our sample strongly differs from the general population of software practitioners.

4.3.3 Data quality

It is hard to assess the correctness of the answers in studies of subjective phenomena. Nevertheless, de Leeuw and van der Zouven have identified five indicators for data quality in telephone and face to face interviews. These indications are [31, p. 286]:

1. Accuracy, or response validity: for this indicator the answer of the respondent is checked against the "true" value as found in official records (e.g. the possession of a drivers license). This indicator is only applicable when validating information is available (...)

2. Absence of S(ocial) D(esirability) Bias; inversely proportional to the number of socially desirable answers on a particular question. An answer is said to be socially desirable when that specific answer is more determined by what is acceptable in society than by the real situation. (. . .)
3. Item response, inversely proportional to the number of "no answer" or "missing data" per question (excluding "do not know" responses)
4. Amount of Information, indicated by the number of responses given in response to an open question or a checklist
5. Similarity of response distributions obtained by different modes. Indicated by no significant difference between the proportions obtained under the different modes. This indicator, though often used, is only a very rough indicator of data quality.

We believe that these indicators could be useful for assessing the quality of our data, even though they were originally created for telephone and face to face interviews. Of the five indicators, four apply to us: (1) is not applicable, as none of our questions involve answers that can be checked against a "true" value as found in an official records.

Doing a very formal and systematic assessment of the data quality is probably unnecessary. We have however, in order to give the reader a short overview, done a quick collation of the data for three of the four remaining indicators. We have also discussed the "Absence of Social Desirability Bias" indicator. Based upon the indicators discussed below, we see no reason to doubt the quality of our data.

Social desirability bias A social desirability bias comes when respondents are asked questions that they rather would not report accurately because of social undesirability. A classic example is the 1989 Virginia gubernatorial election. On election day, all exit polls showed the black candidate, Democrat Douglas Wilder, having a clear lead over his white opponent, Republican Marshall Coleman. Estimates by four major survey organizations showed an advantage to Wilder of 4–11 percentage points, but the actual results gave Wilder the victory by a narrow margin of 6,741 votes of a total of 1.79 million ballots [12]. According to Finkel et al., such errors arise in elections with black and white opposing candidates because "white respondents are more reluctant to report intentions to vote for the white candidate, and more willing to report intentions to vote for the black candidate, than they are to cast their ballots for those candidates on Election Day" [12].

Fowler claims that there is clear evidence that "having respondents answer questions in a self-administered form rather than having an interviewer ask the questions may produce less social desirability bias for some items" [14, p. 99]. We used a self-administered questionnaire for our survey, thus reducing such biases.

Item response Excluding the open ended questions and the questions consisting of a checklist (which are summarized under "Amount of Information", see below), there were very few "no answer" or "missing data". The following table shows the number of "skips" for each question:

Question #	Answered	Skipped
1	113	0
2	113	0
3	112	1
4	109	4
5	110	3
6	106	7
7	113	0

The relative number of "no answers" was at most $\frac{7}{113} = 0.061$. This is very low.

Amount of Information There was one question that consisted of a checklist (Question 8), and two questions that were open ended (questions 9 and 10).

Of the 113 complete responses, 103 ticked one or more items at Question 8. There were 59 participants who answered Question 9, while 59 answered question 10. The response rate at Question 8 ($\frac{103}{113} = 0.91$) is very good, while the response rates at questions 9 and 10 ($\frac{59}{113} = 0.52$) were mediocre. This is in line with research, which shows that there are low response rates for open-ended sections located at the end of structured evaluation forms. See for example Darby, who found that 81.9% of open ended questions were completed at the beginning of questionnaires, 63.0% at the middle, and only 31.6 at the end of questionnaires [6].

Similarity of response distributions We obtained data using two different modes: web collectors and a paper version of the questionnaire. The latter was only done for the javaBin group, as we used a web collector only when we deployed the survey in companies Y and Z. We can thus only look at the javaBin group when assessing this indicator.

We arbitrarily chose three of the questions, and compared the mean m from the two collection modes (see below).

Scale (Question 1)	m, paper	m, web	Difference
A colleague or friend ...	6.23	5.82	0.41
An external expert / guru	4.69	5.12	−0.43
Independent researchers	3.69	3.59	−0.10
Information from the vendor ...	3.31	3.16	0.15
Industrial researchers ...	3.69	3.39	0.30

Response (Question 7)	Percentage, paper	Percentage, web	Difference
Yes	61.5%	52.9%	8.6
No	30.8%	29.4%	1.4
Don't know	7.7%	17.6%	−9.9

Response (Question 8)	**Percentage, paper**	**Percentage, web**	**Difference**
Does not encourage ...	7.7%	18.8%	−11.1
Too busy ...	69.2%	58.3%	10.9
Hard to find	23.1%	45.8%	−22.7
Lack of time	61.8%	39.6%	22.2
Not relevant	15.4%	29.2%	−14.8
Attitudes of colleagues	7.7%	12.5%	−4.8
Attitudes of customers	7.7%	16.7%	−9.0
Other	30.8%	8.3%	22.5

There are no large differences in the distributions of replies to questions one and seven. The differences are somewhat larger when it comes to Question 8. We nevertheless see no reason to doubt the quality of our data.

4.4 Summary

This chapter has described our survey, and explained how we chose a survey as our research method.

We created a survey consisting of 10 questions, and mapped each question to one or more of the SRQs from chapter 1. We described the measures we implemented to minimize possible biases, and the evaluation process that we conducted to make sure that the questions were consistently understood and communicated. We also described two errors we did not catch during the evaluation process; one that possibly is severe, and one that is insignificant.

We described how we administered the questionnaire, and how we used the survey framework called *SurveyMonkey* to deploy it. We deployed the survey at a users' group, and in two Scandinavian consultancy companies. We got 145 respondents, of which 113 completed the survey. In section 4.3.3, we did at quick assertion of the quality of the data we collected.

Now that we have described the survey in this chapter, we will in the next chapter describe how we analyzed the data and the results of our analysis.

Chapter 5

Results

In chapter 4, we discussed why we chose a survey, and how we implemented and administered it. This chapter summarizes the data we collected and shows results from our exploratory modeling.

In section 5.1, we make a case for why we can treat some of our ordinal data as being on a continuous scale. Such a practice is normally discouraged, but we cite studies that suggest that it is relatively unproblematic to use parametric statistics on ordinal data.

Section 5.2 summarizes demographic data from the respondents, and section 5.3 shows the distribution of the answers to each of the 10 questions. Most of the data is also represented graphically, using histograms. The data collected in Question 9 is somewhat special, as classifying it involves some subjectivity; see section 5.3.9.

In section 5.4, we describe a statistical technique called *recursive partitioning*. This technique is good for exploring relationships between variables when one does not have a good prior model. In section 5.4.3, we show how the statistical software *JMP* helps us perform recursive partitioning. Section 5.5 shows 12 cases of recursive partitioning that we have done on our data.

5.1 A note on scales

Basic statistical theory says that one should not treat data on an ordinal scale as being continuous. This is partly because it is hard to quantify the differences between two ordinal values; On an ordinal scale, 8 is more than 4, but not necessarily twice as much. Accordingly, one can not properly calculate a mean value on numbers from an ordinal scale.

However, the practice of treating ordinal scales as being interval scales — scales where one unit on the scale represents the same magnitude across the whole range of the scale — seems widespread. Jakobsson has looked at how ordinal scales were analyzed in nursing journals in 2003 [22]. Of 51 articles that used ordinal scales, only 57% analyzed the data appropriately. Jakobsson notes that even though computing mean values and accompanying standard deviations are "very common" when dealing with ordinal scales, such practices are incorrect.

Nevertheless, there is some justification for treating data from ordinal scales as being continuous. Johnson and Creech have examined the correlation between categorized variables and their continuous analogs [25]. Johnson and Creech claim that while categorization errors (i.e. errors that occur when continuous variables are measured by indicators with only a few categories) to some degree produce distortions in multiple indicator models, "under most of the conditions explored these distortions where not of sufficient magnitude to strongly bias the estimates of the important parameters." This is especially true when five or more categories are used for each question.

Similar results were found by Zumbo and Zimmerman when they used computer simulations to compare results from applying non-parametric *Mann–Whitney U* tests with results from using Student's *t*-test [53]. Zumbo and Zimmerman conclude that

> In general, the results indicate that for statistical hypothesis testing of two-sample location problems (i.e., tests of mean differences) it is not detrimental to use parametric tests on ordinal data. That is, if a mean difference is evident in the latent structure, then the *t*-test or Wilcoxon-Mann-Whitney test performed on data from an ordinal representation will indicate a mean difference, at least as often as a *t*-test on the empirical structure.
>
> (. . .)
>
> Of particular importance is the finding that the power functions for the *t*-tests on the ordinal representations are very similar to the power functions for the nonparametric Wilcoxon–Mann –Whitney test. This indicates that there is no benefit to be gained from excluding the use of parametric statistics on ordinal data. These findings are also true for various sample sizes and significance levels.

Based upon the recommendations of Johnson and Creech as well as Zumbo and Zimmerman and the fact that we used seven categories in our survey, we feel confident that it is unproblematic to treat the scales used in questions 1–6 as continuous.

5.2 Demographic data

The vast majority of the 113 respondents were aged between 25 and 39. 34.5% of the respondents were in the 25 $\rightarrow$ 29 group, while 26.5% were in the 30 $\rightarrow$ 34 group and the same number in the 35 $\rightarrow$ 39 group. See figure 5.1(a) for the full distribution.

All respondents reported that they were working in Norway, with two exceptions: One responded "Company Y", and one responded "Turkey".

For "current position", most respondents (77) reported "developer", while 31 reported "other". The most common answer among those who chose "other" was "system consultant", with variants of "architect" as the second most common. Very few (< 10) reported to be managers. The low number of respondents working in management made us abandon our original strategy of testing "position" as an explanatory variable when doing recursive partitioning, see section 5.4.

The majority of the respondents had 14 years or less of industry experience. 33.6% had between 0 and 4 years, 31.0% had between 5 and 9 years, and 26.5% had between 10 and 14 years. See figure 5.1(b) for the full distribution.

Most respondents (66.4%) had a masters degree or equivalent. 25.7% had a bachelors degree. See figure 5.1(c) for details.

5.3 Distributions

This section shows the distributions of the responses to each question.

All responses are presented in tabular form in the following subsections. Space restrictions forces us to to abridge some of the options to the questions, see Appendix A for the unabridged list. The table columns use the following symbols and abbreviations:

Symbol	Explanation
m	Sample mean
M	Sample median
s	Sample standard deviation
SEM	Standard Error Mean
U 95	Upper limit of the 95% confidence interval
L 95	Lower limit of the 95% confidence interval
N	The number of responses

The sample mean m is the arithmetic mean of the observed values. If a sample consists of the observations $x_1, x_2, \ldots, x_n$, the mean of the observed values is

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

The median M is the midpoint of a distribution. The median is calculated by arranging all observed values in order of size. If the number of observations n is odd, the location of the median M is $\frac{n+1}{2}$ observations up from the bottom. If the number of observations n is even, the median M is the mean of the two center observations.

The sample standard deviation s is a measure of the spread of the values in a sample. s is large if the observations are widely spread about their mean, and small if all of the observations are close to the observed mean.

The sample standard deviation s of n observations with a sample mean of m is

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - m)^2}$$

The sample mean m is an estimate of the mean μ of the population. The sample standard deviation s is an estimate of the standard deviation σ of the population, and s decreases in proportion to the square root of the sample size n:

$$s = \frac{\sigma}{\sqrt{n}}$$

L 95 and U 95 describe the lower and upper limits of a 95% *confidence interval*. Moore and McCabe define a level C confidence interval for a parameter as [35, p. 387]

> an interval computed from sample data by a method that has probability C of producing an interval containing the true value of the parameter

A 95% confidence interval for m is thus the interval that has a probability of 0.95 of containing μ.

Most of the questions have an additional column to the far right, referring to a figure that shows the distribution of the respective responses.

Questions that have a "other (please specify)" field have the responses from these reproduced verbatim directly beneath the aggregated responses. While some of these responses are in Norwegian, the vast majority are in English.

5.3.1 Question 1

"You are considering a new technique or technology in IT development. You have only superficial prior knowledge about this technique or technology. On a scale where 1 is 'not at all' and 7 is 'very much', how important for you is advice from the following persons and entities when you are making your decision? "

Item	m	M	s	SEM	U 95	L 95	N	Figure
Colleague or friend	6.000	6	1.026	0.097	6.191	5.809	113	5.2(a)
Expert	4.991	5	1.228	0.116	5.220	4.762	113	5.2(b)
Independent researchers	3.926	4	1.393	0.131	4.189	3.669	113	5.2(c)
Vendor / supplier	3.230	3	1.282	0.121	3.469	2.991	113	5.2(d)
Industrial researchers	3.566	4	1.231	0.116	3.796	3.337	113	5.2(e)

Other responses

- web search
- utvikler forum, 5

- Private experiments with the technology
- My own prototyping of several of the technologies
- Independently published articles or videos
- Google
- Good documentation, tutorials, rich forums
- General opinion on forums/usergroups
- customer opinion
- blogs/newsgroup postings from people I trust

5.3.2 Question 2

"When looking for knowledge about new development techniques and technologies, to what extent do you consult the following sources?" (1 → 'not at all', 7 → 'very much')

Item	*m*	*M*	*s*	SEM	U 95	L 95	*N*	Figure
Industrial conferences	4.545	5	1.570	0.148	4.839	4.251	112	5.3(a)
Scientific conferences	2.482	2	1.335	0.126	2.732	2.232	112	5.3(b)
Experts, friends	5.991	6	1.081	0.102	6.193	5.790	113	5.3(c)
Popular sc. journals	3.265	3	1.421	0.134	3.530	3.001	113	5.3(d)
Websites, newspapers	4.401	5	1.510	0.142	4.689	4.123	113	5.3(e)
Scientific journals	2.459	2	1.320	0.235	2.708	2.211	111	5.3(f)
Web forums, social media	5.115	5	1.308	0.123	5.359	4.871	113	5.3(e)

Other responses

- The documentation of said technology
- technical websites liker infoq over computerworld!
- podcasts like javaposse
- meetups
- javabin :-)

5.3.3 Question 3

"...On a scale where 1 is 'gut feeling only' and 7 is 'analysis only', how much weight would you give to your analytical findings versus your gut feeling?"

m	*M*	*s*	SEM	U 95	L 95	*N*	Figure
3.920	4	1.164	0.110	4.138	3.702	112	5.4

5.3.4 Question 4

"...On a scale from 1 ('research only') to 7 ('expert advice only'), how would you give weight to the two recommendations?"

m	*M*	*s*	SEM	U 95	L 95	*N*	Figure
4.541	5	1.135	0.109	4.757	4.326	109	5.5

5.3.5 Question 5

"On a scale from 1 ('nothing') to 7 ('very much'), how much do you keep up with research in software engineering?"

m	*M*	*s*	SEM	U 95	L 95	*N*	Figure
3.845	4	1.466	0.140	4.123	3.569	110	5.6

5.3.6 Question 6

"On a scale from 1 ('nothing') to 7 ('very much'), how much confidence do you have in research in software engineering?"

m	*M*	*s*	SEM	U 95	L 95	*N*	Figure
4.104	4	1.187	0.115	4.332	3.875	106	5.7

5.3.7 Question 7

"Do you apply results from research in software engineering in your work?"

Response	*N*	%
Yes	61	53.9%
No	27	23.9%
Don't know	25	22.1%

5.3.8 Question 8

"Do one or more of these barriers prevent you from applying research in your work?"

Barrier	*N*
My organization does not encourage the use of research	13
Too busy meeting immediate goals or deadlines	62
Research results are hard to find	36
Lack of personal time	56
Research is not relevant to my needs	26
Attitudes of colleagues	9
Attitudes or expectations of customers	20

Other reported barriers

- Ridiculously overly complicated result papers. KISS !
- Resarch comes to late i.e. after technology choices have been made and cannot be changed for whatever reasons.
- Reaserching is plainly theorethical, reasearchers often has lack of industry experience
- Lots of research is naivé. 12 years of experience has taught me this. Researchers should not be young people fresh out of school only. We need more researchers with experience in many areas. Of course this is research that can be done without experience but do take this "point" into consideration :-)
- interpretation of the research based on findings in other industries
- I'm very new, so I got enough to learn
- hard to find *relevant* research
- Har ofte andre forutsetninger enn det som kan bevises gjennom research, derfor blir det ofte litt irrelevant
- goes to attitude. reluctance

5.3.9 Question 9

We used ACM's *Computing Classification System* (CCS) for categorizing the responses to Question 9 ("name up to three topics that you feel that software engineering researcher should focus more on") [2]. This is the same classification system that Cai and Card used, see section 3.1.

59 respondents gave at least one topic. 45 gave at least two topics, and 29 gave three topics. This makes the total number of topics given by the respondents $59 + 45 + 29 = 133$. We classified these 133 topics into the categories of section D in the CCS, as this section concerns software engineering. The topics we gathered that did not concern software engineering (example: "learning") were categorized as N/A (not

applicable). Likewise, clearly ambiguous or frivolous replies (examples: “How the real world works”, “High Availebility” [*sic*]) were also categorized as N/A, as well as topics that did not fit into one of the categories (example: “open source”).

As some of the topics given fit into two or more of the CCS categories, there are some trade-offs involved when assigning a category to a topic. We have included all of the collected topics verbatim and our accompanying assessments in appendix B, so that the reader may verify our categorizations.

The raw count of the categorized topics are presented below and in figure 5.9.

Category	*n*
D.2.1 Requirements/Specifications	4
D.2.2 Design Tools and Techniques	1
D.2.3 Coding Tools and Techniques	11
D.2.4 Software/Program Verification	6
D.2.5 Testing and Debugging	5
D.2.6 Programming Environments	2
D.2.7 Distribution, Maintenance, and Enhancement	7
D.2.8 Metrics	1
D.2.9 Management	25
D.2.10 Design	1
D.2.11 Software Architecture	5
D.2.12 Interoperability	5
D.2.13 Reusable Software	0
N/A	60
$\sum$	133

5.3.10 Question 10

“Which software engineering activity or job task is your main daily focus at present?”

This question was included for the purpose of controlling that the the topics given in Question 9 were not the same as the respondents’ current job task, as it is possible that respondents overreport topics that they are working on at present.

Most responses from this question can be found in appendix B, where they are grouped with the responses from Question 9. Responses to Question 10 from respondents who did not give any topics in Question 9 are not shown, as these are unimportant.

We will not discuss the responses to Question 10 further in chapter 6, but a quick, informal analysis suggests that the correlation between the responses to questions 9 and 10 is surprisingly low.

5.4 Exploratory Modeling and JMP

Exploratory modeling is the process of using an automatic method to explore large amounts of data, to find patterns and discoveries [40, p. 457].

There are several exploratory modeling techniques that one may choose among, such as *recursive partitioning* and *neural nets*. We chose the former, as we have prior experience with this technique.

Similarly, there is a wide array of different statistical applications we could use for doing this. Some of the most widely known are *SAS* and *JMP*. Even though the former is more powerful, we chose the latter. The main benefits of *JMP* is that it is relatively cost-efficient (both in terms of licensing and computing power), as well as easy to learn.

5.4.1 Recursive partitioning

JMP has a feature called the *Partition platform*. This feature recursively partitions a data set, and is advantageous for exploring relationships when one does not have a good prior model. The technique is very similar to techniques known as CART, CHAID, and C4.5 [40, p. 458].

The data is partitioned according to the relationship between X and Y values, creating a tree of partitions. Y values are *response variables*, while X values are *explanatory variables* (or "factors", as *JMP* calls them). Recall the definition of these terms: (from Moore and McCabe [35, p. 103])

> **Response variable**: measures an outcome of a study
> **Explanatory variable**: explains or causes changes in the response variables

Recursive partitioning works with both nominal and ordinal scales (categorical data), and also with continuous scales. Although most of our scales were ordinal, we chose to treat them as being continuous. See section 5.1 for more about this.

For continuous Xs, the partitions (splits) are created by a *cutting value*. The sample is divided into values below and above this value. For continuous Ys, the platform fits means, and creates splits that separate the means by examining the sums of squares due to the mean differences. The split is chosen so that the the difference in the responses between the two values are maximized. The *Partition platform* examines a very large number of possible splits, and chooses the optimal (i.e. the most significant of the alternatives) [42, p. 657].

5.4.2 LogWorth

An index called *LogWorth* is used as a significance measure for each split. LogWorth is the negative log of the adjusted p-value:

$$\text{LogWorth} = -\log_{10} p\text{-}value$$

According to the JMP manual, [42, p. 667]

> [t]he adjusted log–p–value is calculated in a complex manner that takes into account the number of different ways a split can occur. This calculation is very fair compared to the unadjusted p-value, which is biased to favor X's with many levels, and the Bonferroni p-value, which overadjusts to the effects that it is biased to favor — those X's with small number of levels.

This value is reported on a logarithmic scale to avoid values that underflow in machine floating–point form, and as a negative `log` so that large values as associated with significant terms.

p-values less than 0.05 are reflected by LogWorth-values greater than 1.30. See Hannay et al. for details [20].

5.4.3 Partitioning examples

Figure 5.10 shows an example of recursive partitioning in *JMP*. In this example, we have used question six — "How much confidence do you have in research in software engineering?") — as X (the explanatory variable), and question 7 — "do you apply research in software engineering in your work?" — as Y (the response variable). (1) shows the raw count of Y (excluding the "don't know" responses). 69.3% of the respondents claim to use research in their work, while 30.7% claim *not* to use research. There were 88 responses.

These 88 responses were then split into (2) and (3). (2) shows the respondents with a confidence in research of ≥ 4, while (3) shows the respondents who had a confidence in research < 4. Among group (2), 79.4% claimed to use research in their work, while 20.6% claimed not to. In group (3), the corresponding numbers were 44.0% and 56.0%.

(4) shows the LogWorth value for the split into (2) and (3). As this value is > 1.30, the difference between groups (2) and (3) is significant ($P < 0.05$). On the other hand, the LogWorth values at (5) and (6) are ≤ 1.30, and further partitioning will therefore *not* lead to significant results. If one of either (5) or (6) were > 1.30, we would have spilt the corresponding group further into two subgroups, and recursively continued splitting until there were no more significant differences — i.e. when no more LogWorth values were > 1.30.

The example in figure 5.10 had nominal data as the Y variable. Ordinal Ys are handled the same way. *JMP* does recursive partitioning slightly different on continuous data. Figure 5.11 concerns partitioning of a continuous Y, in this example the testing framework in Question 4 ("on a scale from 1 [research only] to 7 [expert advice only] how would you give weight to the recommendations?"). The "scientific conferences" source from Question 2 is used as X.

(7) shows that there were 109 responses to the "scientific conferences" source in question 2, with $m = 4.54$ and $s = 1.13$. This group was split into (8) and (9). As $LogWorth > 1.30$, the difference between (8) and (9) is significant ($P < 0.05$). (8) has $m = 3.72$, while (9) has $m = 4.63$. Note that the reported value "Difference" to the right of (7) is the difference between the mean values of (8) and (9). Respondents who report to use scientific conferences ≥ 5 when gaining knowledge about new development techniques are thus shifted 0.905 in favor of giving weight to research when choosing new technologies, in comparison to those who use scientific conferences < 5 when learning about new development techniques.

5.5 Partitioning our data

It might be tempting to do recursive partitioning on *all* questions and items, as the number of candidate response variables is not overwhelming. However, time constraints forced us to select 12 interesting scenarios to perform this task on. These scenarios are described in the subsections below, each having a rationale for its inclusion.

For all partitioning, we used the "Maximize Significance" setting, set "Missing Value Rule" to "Random", and "Minimum Size Split" to 5.

The upper table in each subsection concerns the Y (response variable), and lists the observed mean m_Y, sample standard deviation s_Y, and the number of responses N_Y.

The tables directly below show the LogWorth values for each of the response variables. The response variables are denoted by $X_1, \ldots, X_n$. For each significant partitioning that it is possible to make ($LogWorth > 1.30$, $P < 0.05$), there is an additional table showing the relevant characteristics of the partitions. These partitions are denoted as X_{na} and X_{nb}, where n refers to the X in question. Example: If a variable X_5 is split, the partitions are denoted by X_{5a} and X_{5b}.

5.5.1 Y: Confidence in research, X: all items from Question 1

Y : **Question 6** — How much confidence do you have in research?
$X_{1,2,3,4,5}$:**Question 1** — How important to you is the advice of the following [when considering technologies]?

This scenario was chosen because we suspect a relationship between the confidence practitioners have in research, and how they rate the advice of researchers when choosing between technologies.

m_Y	s_Y	N_Y
4.104	1.187	106

X	Item, question 1	LogWorth
X_1	Colleague or friend	0.02
X_2	Expert	0.09
X_3	Independent researchers	0.35
X_4	Vendor / supplier	0.23
X_5	Industrial researchers	0.45

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.2 Y: Confidence in research, X: all items from Question 2

Y : **Question 6** — How much confidence do you have in research?
$X_{1,2,3,4,5,6,7}$:**Question 2** — When looking for knowledge, [. . .] to what extent do you consult the following sources?

This scenario was chosen because we suspect a relationship between the confidence practitioners have in research, and how they look for knowledge.

m_Y	s_Y	N_Y
4.104	1.187	106

X	Item, question 2	LogWorth
X_1	Industrial conferences	0.34
X_2	Scientific conferences	0.48
X_3	Experts, friends	1.53
X_4	Popular sc. journals	0.02
X_5	Websites, newspapers	0.63
X_6	Scientific journals	1.61
X_7	Web forums, social media	0.84

X_3 and X_6 have LogWorth values > 1.30. We will therefore partition these variables:

X	N	m_Y	s_Y
X_{3a}: Experts, friends < 5	10	3.20	0.79
X_{3b}: Experts, friends ≥ 5	96	4.20	1.18

X	N	m_Y	s_Y
X_{6a}: Scientific journals < 4	84	3.94	1.20
X_{6b}: Scientific journals ≥ 4	22	4.72	0.94

There is thus a significant ($P < 0.05$) difference when it comes to confidence in research in software engineering between those who give less than 5 to "Experts, friends" and those who give 5 or more. The difference is 1.00. Likewise, there is a significant difference between those who give less than 4 to scientific journals and those who give four or more, the difference being 0.76 in favor of the latter group.

5.5.3 Y: Apply research in work, X: keep up with and confidence in research

Y : **Question 7** — Do you apply results from research in software engineering in your work?
$X_{1,2}$: **Question 5** — How much do you keep up with research?, and **Question 6** — How much confidence do you have in research?

This scenario was chosen because we suspect that practitioners who keep up with and have confidence in research might be more inclined to apply research in their work.

N_{Yes}	N_{No}	$N_{Dont\ know}$
61	27	25

X	Question	LogWorth
X_1	Keep up with research	9.95
X_2	Confidence in research	1.74

Both LogWorth values are > 1.30, and we partition:

X	N_{Yes}	N_{No}	$N_{Dont\ know}$	$\%_{Yes}$	$\%_{No}$	$\%_{Dont\ know}$
X_{1a}: Keep up with research ≥ 4	51	10	4	78.5	15.4	6.1
X_{1b}: Keep up with research < 4	10	17	21	20.8	35.4	43.8

X	N_{Yes}	N_{No}	$N_{Dont\ know}$	$\%_{Yes}$	$\%_{No}$	$\%_{Dont\ know}$
X_{2a}: Confidence in research < 4	10	14	6	33.3	46.7	20.0
X_{2b}: Confidence in research ≥ 4	51	13	19	61.4	15.7	22.9

It is a significant ($P < 0.05$) difference between practitioners who report to keep up with research ≥ 4 and practitioners who report to keep up with research < 4. Among the former group, 78.5% claim to apply research in their work, while only 20.8% of the latter group do so.

Similar results are found when it comes to those who have a confidence in research ≥ 4; in this group, 61.4% apply results from research in their work. Among those with less confidence in research (< 4), only 33.3% claim to apply results from research in their work.

5.5.4 Y: Recommendations for testing framework, X: barriers to applying research

Y : **Question 4** — On a scale from 1 ("research only") to 7 ("expert advice only"), how much weight would you give to recommendations [when considering a testing framework]?
$X_{1,2,3,4,5,6,7}$: **Question 8** — Do one or more of these barriers prevent you

from applying research in your work?

This scenario was chosen because practitioners who claim that barriers prevent them from applying research could be more likely to value experts' advice over research.

m_Y	s_Y	N_Y
4.541	1.135	109

X	**Item, question 8**	**LogWorth**
X_1	My organization does not encourage the use of research	0.04
X_2	Too busy meeting immediate goals or deadlines	0.32
X_3	Research results are hard to find	0.12
X_4	Lack of personal time	0.07
X_5	Research is not relevant to my needs	0.68
X_6	Attitudes of colleagues	0.76
X_7	Attitudes or expectations of customers	0.19

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.5 Y: Renewal of process model, X: barriers to applying research

Y : **Question 3** — On a scale from 1 (“gut feeling only”) to 7 (“analysis only”), how much weight would you give to analytical findings versus gut feeling [when considering a renewed process model]?
$X_{1,2,3,4,5,6,7}$: **Question 8** — Do one or more of these barriers prevent you from applying research in your work?

This scenario was chosen because practitioners who claim that barriers prevent them from applying research could be more likely to value gut feeling over research/analysis.

m_Y	s_Y	N_Y
3.920	1.164	112

X	**Item, question 8**	**LogWorth**
X_1	My organization does not encourage the use of research	0.64
X_2	Too busy meeting immediate goals or deadlines	0.81
X_3	Research results are hard to find	0.11
X_4	Lack of personal time	0.22
X_5	Research is not relevant to my needs	0.06
X_6	Attitudes of colleagues	1.89
X_7	Attitudes or expectations of customers	0.12

X_6 has a LogWorth value > 1.30. We will therefore partition this variable:

X	**N**	**m_Y**	**s_Y**
X_{6a}: Yes, the attitudes of colleagues is a barrier	9	3.00	0.87
X_{6b}: No, the attitudes of colleagues is not a barrier	103	4.00	1.15

It is a significant ($P < 0.05$) difference between practitioners who report that the attitudes of colleagues *is* a barrier when applying research, and those who report that this *is not* a barrier. Practitioners in the first group are shifted in favor of using gut feeling when deciding on a new process model, while the latter group is shifted towards using analysis.

5.5.6 Y: Relying on colleagues and friends when making decisions, X: all items from Question 2

Y : **Question 1a** — On a scale from 1 ("not at all") to 7 ("very much"), how important to you is advice from [a colleague or friend] when you are considering a new technique or technology?
$X_{1,2,3,4,5,6,7}$:**Question 2** — When looking for knowledge, [. . .] to what extent do you consult the following sources?

This scenario was chosen because we supect a clear relation between which sources practitioners use when looking for knowledge about new development techniques, and how they value the advices of the said sources when making decisions.

m_Y	**s_Y**	**N_Y**
6.00	1.03	113

X	**Item, question 2**	**LogWorth**
X_1	Industrial conferences	1.76
X_2	Scientific conferences	0.14
X_3	Experts, friends	7.36
X_4	Popular sc. journals	0.03
X_5	Websites, newspapers	0.44
X_6	Scientific journals	0.39
X_7	Web forums, social media	1.01

X_1 and X_3 have LogWorth values > 1.30, we thus partition these variables:

X	**N**	**m_Y**	**s_Y**
X_{1a}: Industrial conferences < 3	17	5.35	1.66
X_{1b}: Industrial conferences ≥ 3	96	6.11	0.83

X	**N**	**m_Y**	**s_Y**
X_{3a}: Experts, friends < 7	66	5.61	1.01
X_{3b}: Experts, friends $= 7$	47	6.55	0.77

Practitioners who look for knowledge at industrial conferences ≥ 3 rely more on friends and colleagues when making decisions than practitioners who look for knowledge at industrial conferences < 3. The former group values advice from friends and colleagues at $m = 6.55$, while the latter values advice from friends and colleagues at $m = 5.61$. This difference is significant ($P < 0.05$).

Practitioners who consult experts and friends $= 7$ when looking for knowledge consider the advice of colleagues and friends more important than practitioners who consult experts and friends < 7. The former group has a mean importance of advice of colleagues and friends at 6.55, while the latter group assigns a mean importance of 5.61. The difference is significant ($P < 0.05$).

5.5.7 Y: Apply research in work, X: age, education, and years of industry experience

Y : **Question 7** — Do you apply results from research in software engineering in your work?
$X_{1,2,3}$: Age, highest level of formal education, and years of industry experience

This scenario was chosen because demographic characteristics could influence whether practitioners apply research.

N_{Yes}	N_{No}	$N_{Dont\ know}$
61	27	25

X	**Attribute**	**LogWorth**
X_1	Age	0.18
X_2	Level of formal education	0.12
X_3	Industry experience	0.11

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.8 Y: Confidence in research, X: age, education, and years of industry experience

Y : **Question 6** — How much confidence do you have in research?
$X_{1,2,3}$: Age, highest level of formal education, and years of industry experience

This scenario was chosen because demographic characteristics could influence practitioners' confidence in research.

m_Y	s_Y	N_Y
4.104	1.187	106

X	Attribute	LogWorth
X_1	Age	0.77
X_2	Level of formal education	0.08
X_3	Industry experience	0.20

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.9 Y: Renewal of process model, X: age, education, and years of industry experience

Y : **Question 3** — On a scale from 1 ("gut feeling only") to 7 ("analysis only"), how much weight would you give to analytical findings versus gut feeling [when considering a renewed process model]?
$X_{1,2,3}$: Age, highest level of formal education, and years of industry experience

This scenario was chosen because demographic characteristics could influence how practitioners give weight to analysis versus gut feeling.

m_Y	s_Y	N_Y
3.920	1.164	112

X	Attribute	LogWorth
X_1	Age	0.61
X_2	Level of formal education	0.45
X_3	Industry experience	0.14

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.10 Y: Recommendations for testing framework, X: age, education, and years of industry experience

Y : **Question 4** — On a scale from 1 ("research only") to 7 ("expert advice only"), how much weight would you give to recommendations [when considering a testing framework]?
$X_{1,2,3}$: Age, highest level of formal education, and years of industry experience

This scenario was chosen because demographic characteristics could influence how practitioners give wight to research versus expert advice.

m_Y	s_Y	N_Y
4.541	1.135	109

X	Attribute	LogWorth
X_1	Age	0.09
X_2	Level of formal education	0.15
X_3	Industry experience	0.38

All LogWorth values are ≤ 1.30. It is thus not possible to partition any of the Xs in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.11 Y: Recommendations for testing framework, X: Renewal of process model

Y : **Question 4** — On a scale from 1 (“research only”) to 7 (“expert advice only”), how much weight would you give to recommendations [when considering a testing framework]?
X_1 : **Question 3** — On a scale from 1 (“gut feeling only”) to 7 (“analysis only”), how much weight would you give to analytical findings versus gut feeling [when considering a renewed process model]?

This scenario was chosen because we suspect a clear connection between how practitioners choose between analysis and gut feeling and how they choose between research and expert advice.

m_Y	s_Y	N_Y
4.541	1.135	109

X	LogWorth
X_1	1.29

The LogWorth value is ≤ 1.30. It is thus not possible to partition X_1 in a way that creates a significant ($P < 0.05$) difference between two partitions.

5.5.12 Y: Keeping up with research, X: Collector

Y : **Question 5** — How much do you keep up with research in software engineering?
X_1 : Collectors (i.e. Company Y, Company Z, and javaBin)

This scenario was chosen as different organizational cultures may influence how the employees keep up with research.

m_Y	s_Y	N_Y
3.845	1.466	110

X	**LogWorth**
X_1	0.26

The LogWorth value is ≤ 1.30. It is thus not possible to partition X_1 in a way that creates a significant ($P < 0.05$) difference between collectors.

5.6 Summary

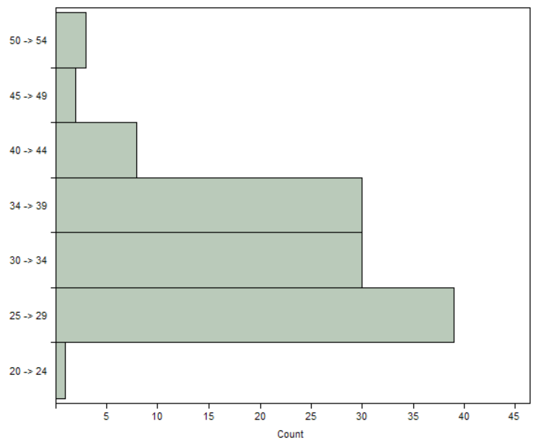
In section 5.1, we discussed scales, and showed why data that strictly speaking is on an ordinal scale can be treated as being on a continuous scale.

Section 5.2 summarizes the demographic distribution of our respondents. Most respondents were relatively young (age < 40) and had less than 15 years of industry experience. Virtually everybody worked in Norway, and the overwhelming majority reported to be developers. Less than 10 respondents worked in management.

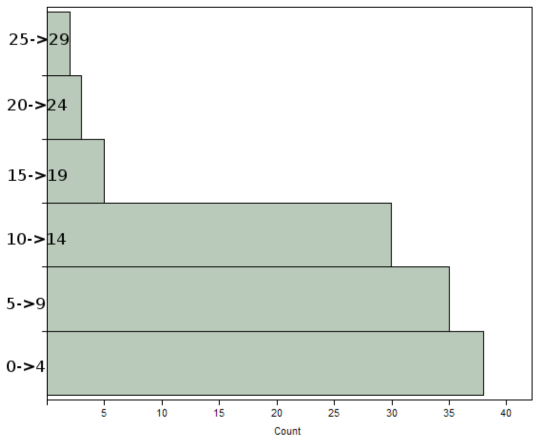
Section 5.3 begins by defining some key concepts of statistics, such as sample means, sample standard deviations, and confidence intervals. We then described the data distributions from the ten questions, and also reported the replies in the "other" fields. For Question 9, we also categorized the topics we collected into ACM's Computing Classification System, so that we in the next chapter can compare these with Cai and Card's findings discussed in chapter 3.

We described exploratory modeling, recursive partitioning, and the software package JMP in the first subsections of section 5.4, and gave an example of how to do partitioning in section 5.4.3. In section 5.5, we described 12 partitioning scenarios. The majority of these scenarios involved several explanatory variables. It was possible to do statistically significant splits on seven explanatory variables spanning four scenarios.

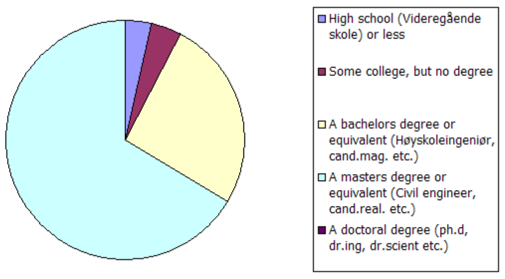
Now that we have reported the distributions of all questions and done a statistical analysis of our data, we are ready to discuss the implications of our findings in chapter 6.



(a) Grouped by age

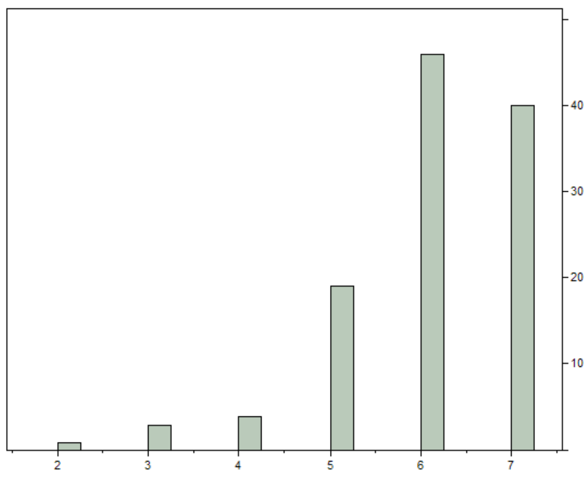


(b) Grouped by industry experience (years)

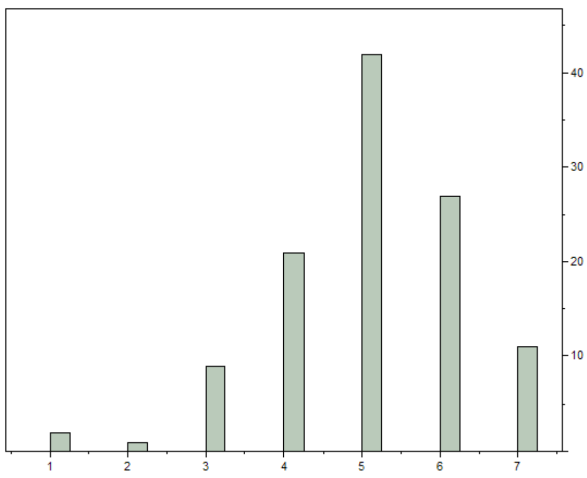


(c) Grouped by highest level of education

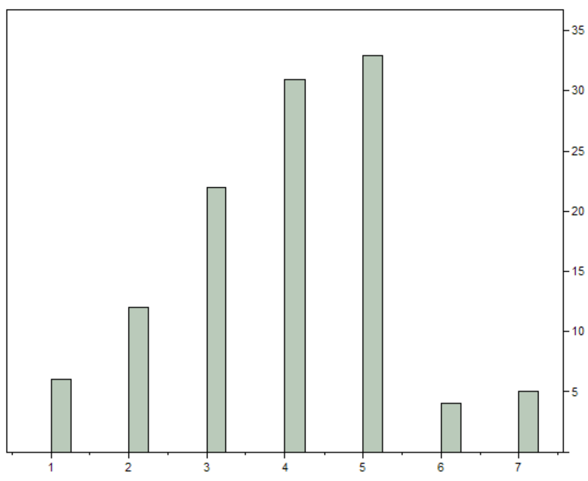
Figure 5.1: Demographical distribution of the respondents



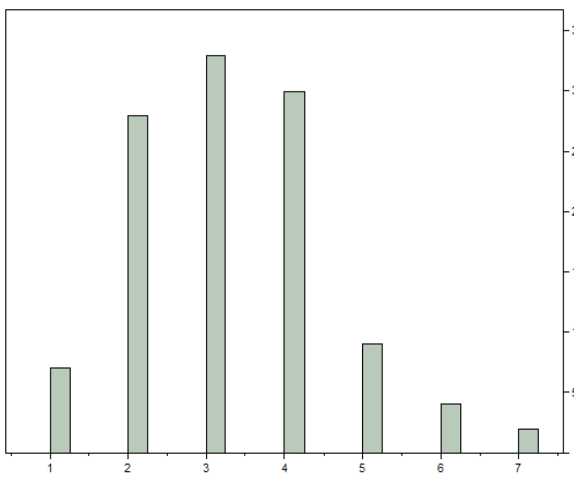
(a) "A colleague or good friend with experience with this technique"



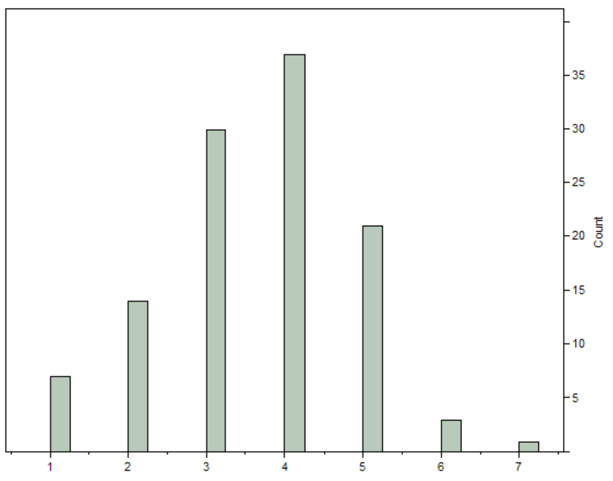
(b) "An external exert / guru"



(c) "Independent researchers"

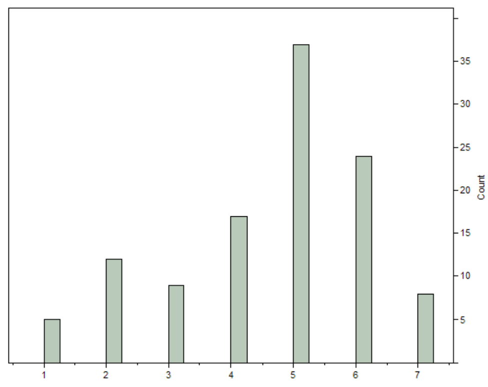


(d) "Information from the vendor or supplier"

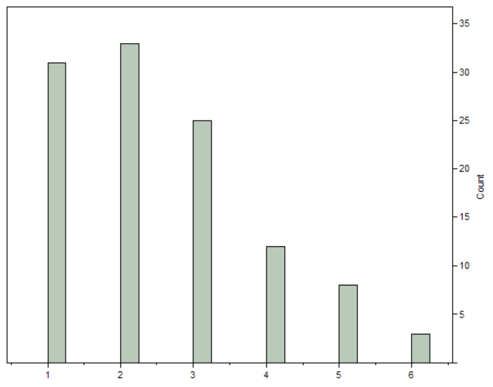


(e) "Industrial researchers"

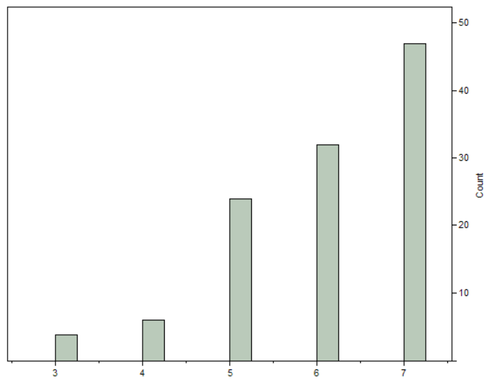
Figure 5.2: Histograms of the responses, Question 1



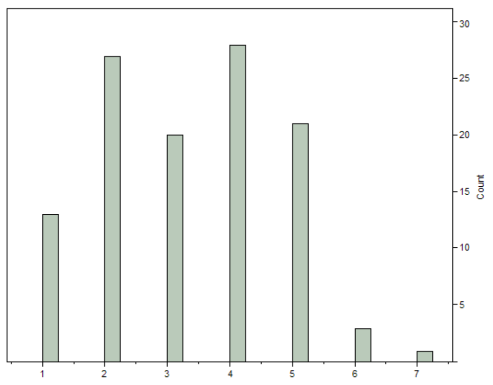
(a) "Industrial conferences"



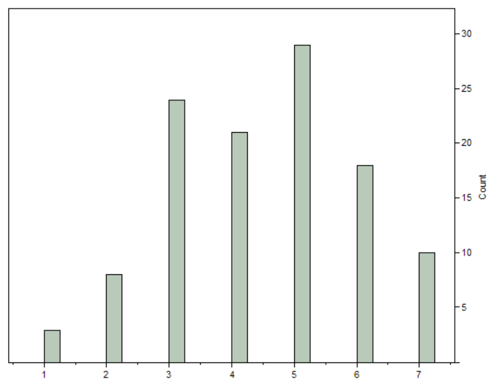
(b) "Scientific conferences"



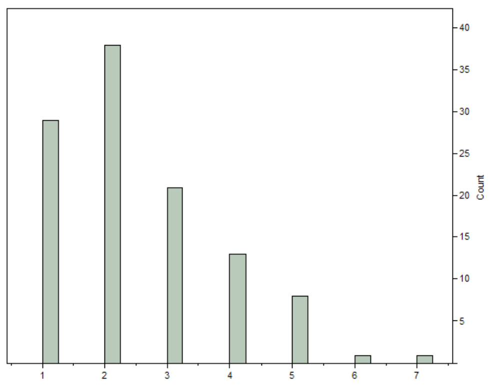
(c) "Experts, gurus, colleagues, and friends in the IT industry"



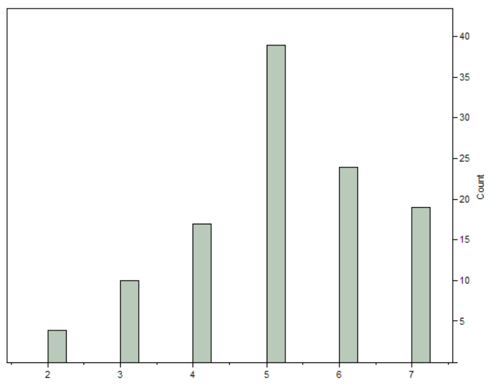
(d) "Popular scientific journals"



(e) "Websites and IT newspapers"



(f) "Scientific journals"



(g) "Web forums, mailing lists, blogs, and other social media"

Figure 5.3: Histograms of the responses, Question 2

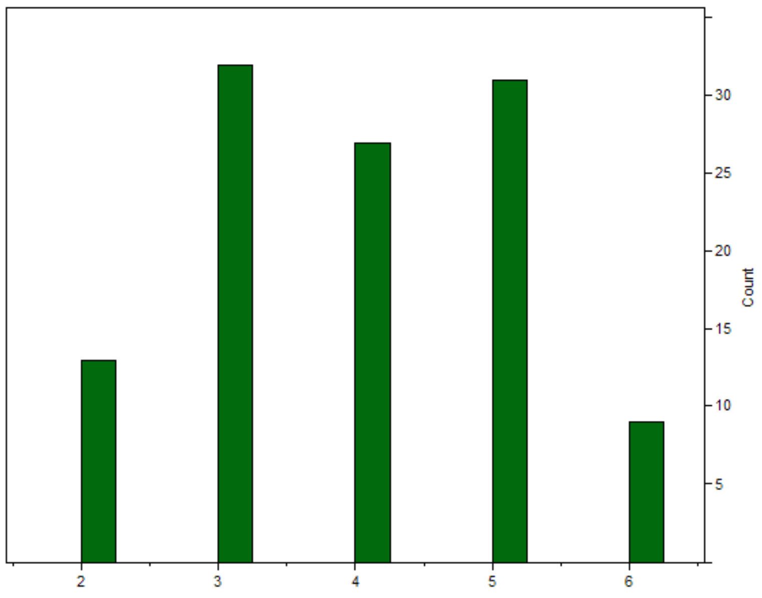


Figure 5.4: Histogram of the responses from Question 3.

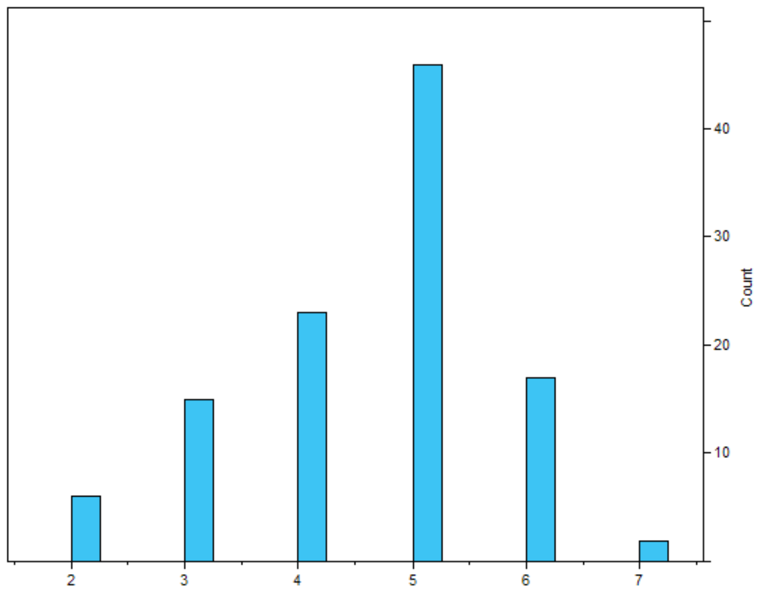


Figure 5.5: Histogram of the responses from Question 4.

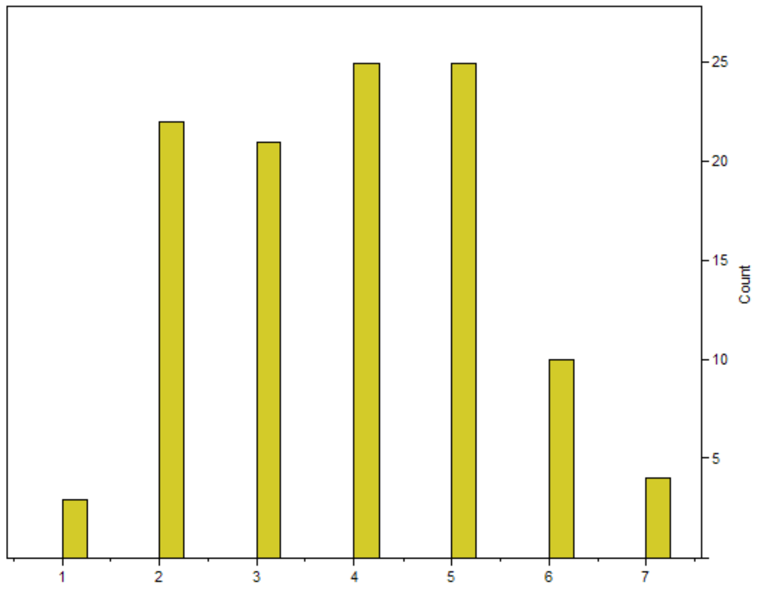


Figure 5.6: Histogram of the responses from Question 5.

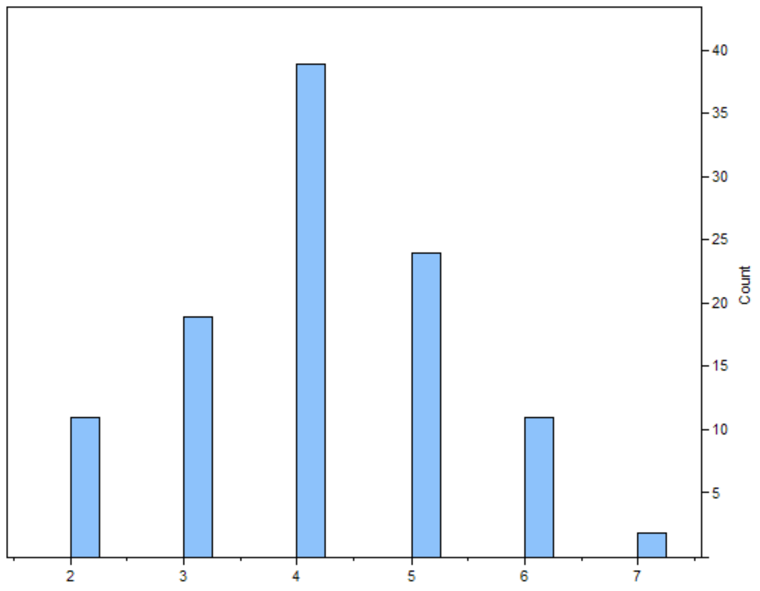


Figure 5.7: Histogram of the responses from Question 6.

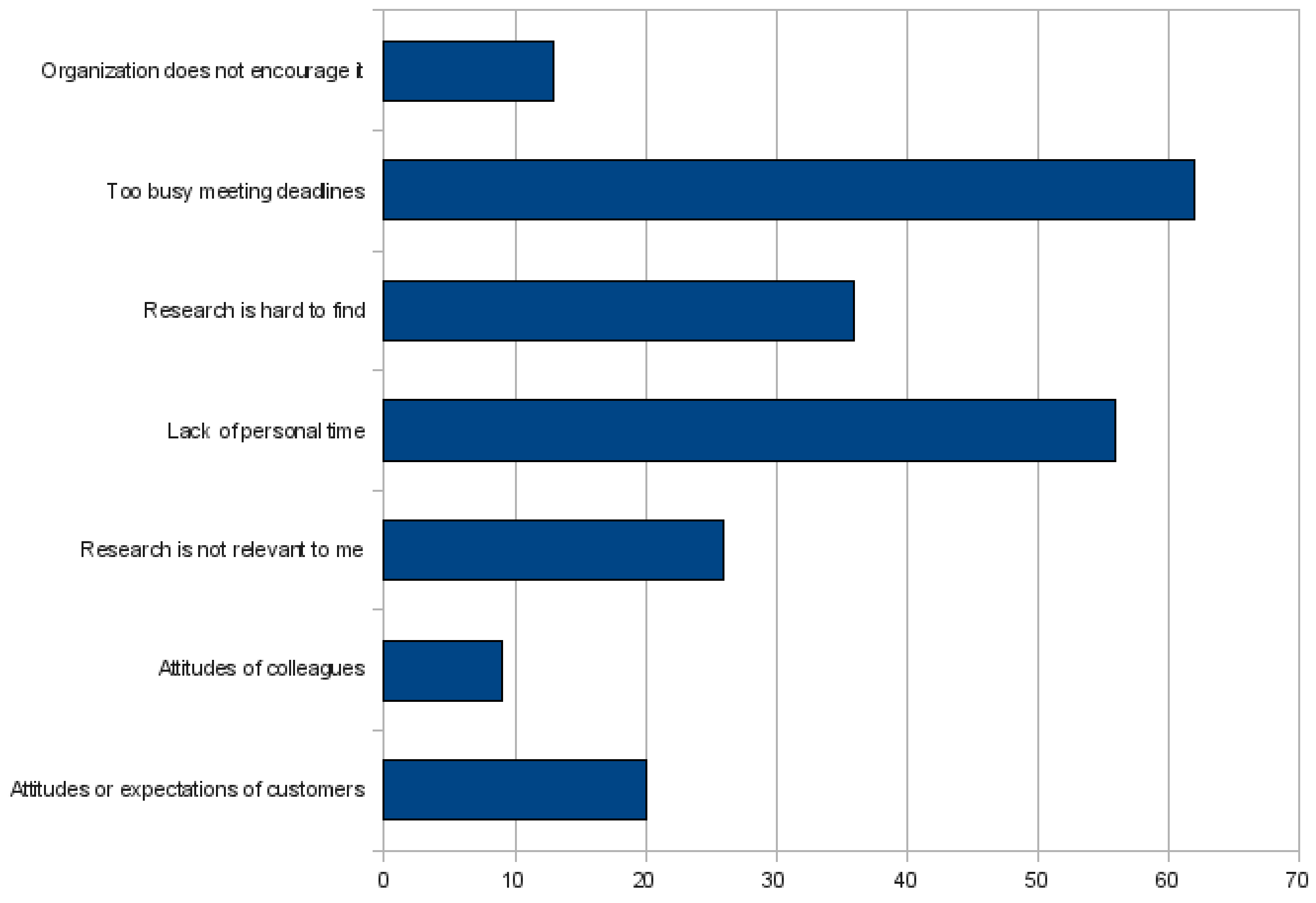


Figure 5.8: The barriers reported in Question 8.

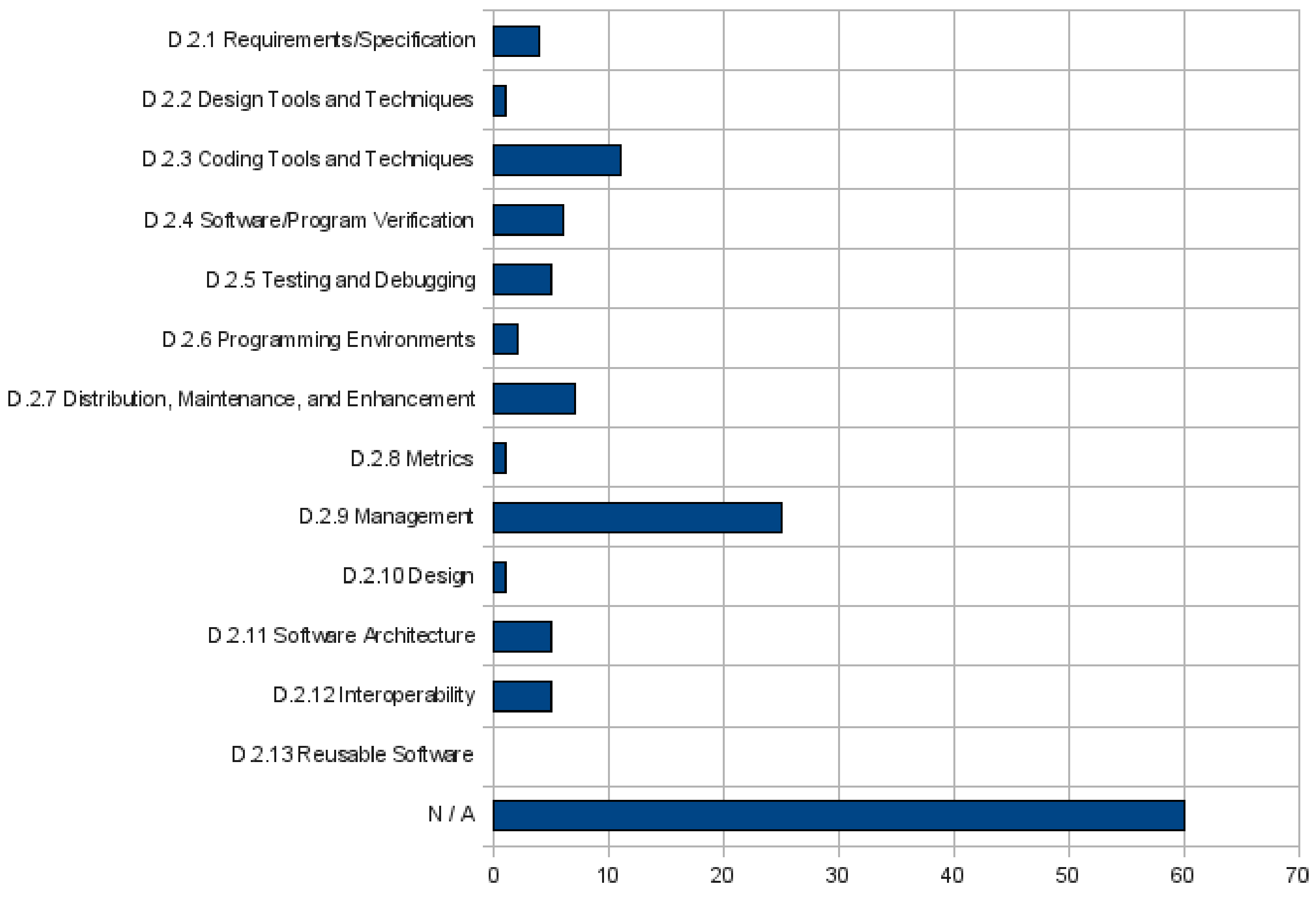
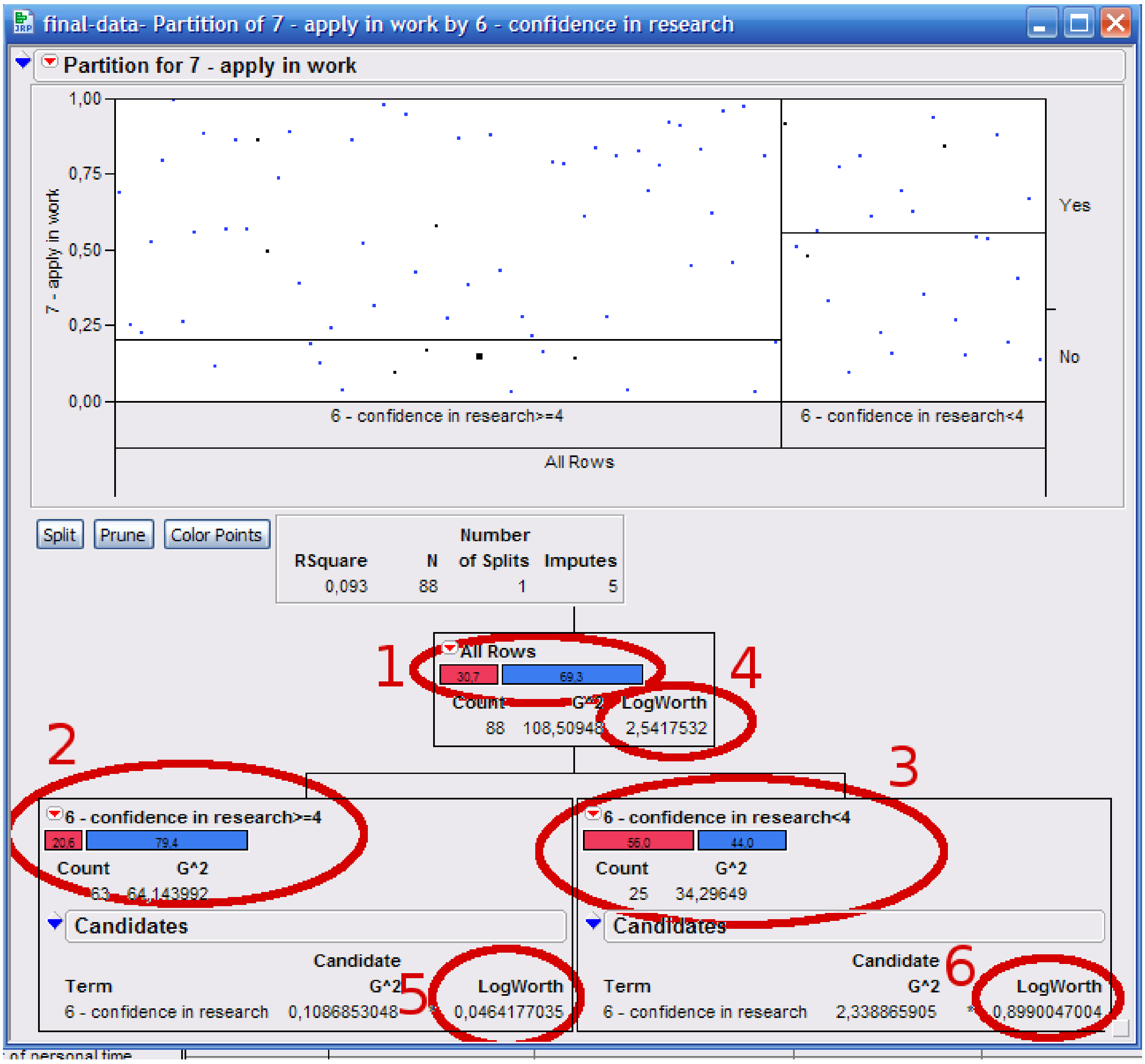


Figure 5.9: The topics from Question 9 categorized using the CCS system.



final-data- Partition of 7 - apply in work by 6 - confidence in research

Partition for 7 - apply in work

RSquare	N	Number of Splits	Imputes
0,093	88	1	5

All Rows

Count	G^2	LogWorth
88	108,50948	2,5417532

6 - confidence in research>=4

Count	G^2
63	64,143992

Candidates

Term	Candidate G^2	LogWorth
6 - confidence in research	0,1086853048	0,0464177035

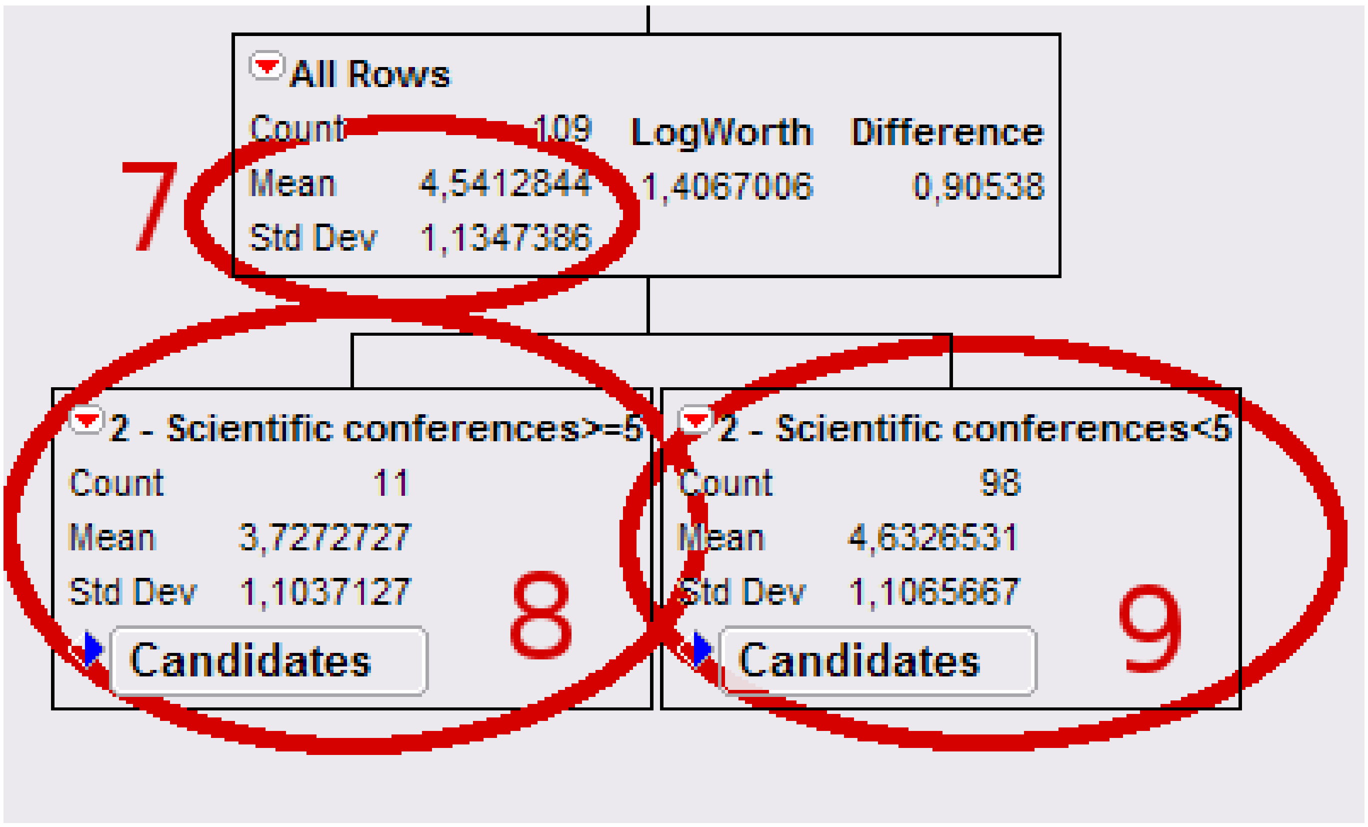
6 - confidence in research<4

Count	G^2
25	34,29649

Candidates

Term	Candidate G^2	LogWorth
6 - confidence in research	2,338865905	* 0,8990047004

Figure 5.10: Example of recursive partitioning in *JMP*, nominal data. See section 5.4.3 for annotations.



```
All Rows
Count        109   LogWorth   Difference
Mean   4,5412844   1,4067006     0,90538
Std Dev 1,1347386
```

```
2 - Scientific conferences>=5
Count          11
Mean    3,7272727
Std Dev 1,1037127
```

```
2 - Scientific conferences<5
Count          98
Mean    4,6326531
Std Dev 1,1065667
```

Figure 5.11: Another example of recursive partitioning, this time on continuous data. See section 5.4.3 for annotations.

Chapter 6

Discussion

In the previous chapter, we presented the results of our survey. In this chapter, we summarize the key points of our results, and discuss the implications of these findings. To help us, we use the findings from the research papers discussed in chapter 3.

Our four SRQs are discussed individually in the four subsections of section 6.1.

Section 6.2 discusses the four relevant kinds of validity threats, with threats to construct validity (section 6.2.1) and threats to the external validity (section 6.2.3) as the most important.

We used ethical guidelines aimed at social researchers when conducting our survey; these are shown in section 6.3. We then discuss how we have complied with these guidelines.

6.1 How practitioners value research

The research question (RQ) of this thesis concerns practitioners' attitudes to research:

> **RQ** How do software practitioners value research when making decisions?

As described in section 1.2, the RQ was further divided into four specific research questions (SRQs). These four SRQs are discussed separately in the following subsections.

6.1.1 Learning about new techniques and technologies

> **SRQ1** How do software practitioners learn about new techniques and technologies?

This SRQ was addressed in Question 2 of the questionnaire. The by far mostly used source was experts and friends, with $m = 5.99$. Other much used sources were web forums, mailing lists, blogs and other social media, with $m = 5.12$. Websites and newspapers scored $m = 4.40$, while industrial conferences scored $m = 4.55$. Popular scientific journals had a mean of $m = 3.27$. The two least used sources were scientific conferences and scientific journals, with $m = 2.48$ and $m = 2.46$ respectively.

It did not surprise us that scientific conferences and scientific journals were the two least used sources of knowledge about new development techniques and technologies. The exploratory analysis (see section 5.5.3) gave an interesting insight: Practitioners who use scientific journals a lot to gain knowledge have more confidence in research than practitioners who do not use scientific journals that much when looking for knowledge: The respondents who gave < 4 to scientific journals as a source of knowledge have a mean confidence in research of 3.94, while those who gave scientific journals ≥ 4 have a mean confidence in research of 4.72. The difference is significant ($P < 0.05$). It thus looks like reading about research gives confidence in it — something which should be reassuring to researchers.

Unsurprisingly, there was a high correlation between those who sought advice from experts and friends when making decisions, and those who to a great extent consult friends and experts when learning about new technologies (see section 5.5.6).

6.1.2 Expert advice contra research

> **SRQ2** Do software practitioners value experts' advice more than they value research?

This SRQ was perhaps the most important to us, and several of the questions of the questionnaire were related to it.

When considering a new technology Question 1 asked the respondents how they valued the advice of colleagues, friends, experts, independent- and industrial researchers, and the vendor or supplier when making a technology decision. It turned out that the advices of colleagues and friends were the most important, with $m = 6.00$. Information from the vendor or supplier was the least important, with $m = 3.23$. This seems rational, as friends and colleagues supposedly are neutral, while the vendors or suppliers will have a strong interest in the practitioner making a particular decision. For experts, the sampled mean was $m = 4.99$.

Somewhat more surprising was the fact that the respondents did not see much of a difference between independent researchers and industrial researchers, having $m = 3.93$ and $m = 3.57$ respectively. Industrial researchers will often have strong ties to commercial interests, and we would hope that practitioners found them considerably less trustworthy than independent (i.e. academic) researchers. However, the wide use of untrustworthy numbers from for example the *Chaos Report* (briefly discussed in section 2.2) do support the notion of practitioners who are unable to differentiate between trustworthy and untrustworthy research.

When choosing between analysis, gut feeling, and expert advice Choosing between research and expert advice was the issue in Question 4. On a scale where 1 was "research only" and 7 "expert advice only", the mean reply was 4.54. Practitioners are thus valuing research and experts' advice about equally, with a small shift towards experts' advice.

Question 3 concerned giving weight to analysis versus gut feeling (intuition) when making a decision. As noted in our questionnaire specification (see section 4.2), we consider this question related to Question 4 and SRQ 2. The result from this question was $m = 3.92$, which is about halfway between "gut feeling only" (1) and "analysis only" (7).

From questions 3 and 4, it seems clear that practitioners value intuition, experts' advice, and analysis/research about equally when making decisions. This is somewhat surprising, as experts were reported to be considerably more used than research when practitioners learnt about new technologies.

There is an apparent incompatibility between the results from Question 1 and the results from questions 3 and 4. While questions 3 and 4 showed little difference between experts' advice, gut feeling, and analysis, Question 1 showed that respondents valued experts more than research. There are however some important differences between the premises of the questions; in Question 1, the respondents were asked to rank advices when they had **only superficial prior knowledge** about the subject, while in questions 3 and 4, the respondents were told that they **made the analysis themselves**. It could thus be that practitioners value research / analysis more when they have assessed it themselves than they do when they more or less passively are advised by researchers.

The practitioners who claim that the attitudes of colleagues create a barrier to using research are more prone to using gut feeling over analysis. This was shown in section 5.5.5. The difference, which was significant ($P < 0.05$), was quite large. Practitioners reporting that colleagues created a barrier had $m = 3.00$ on the gut feeling / analysis question when choosing a new process model, while those not reporting colleagues as a barrier had $m = 4.00$.

We discussed intuition in section 3.3, and cited Tversky and Kahneman and their findings that "acquaintance with formal logic and with probability theory does not extinguish erroneous intuitions." ICT practitioners are probably familiar with logic (either through formal logic or through practical experience with algorithms), but if Tversky and Kahneman are right, this does not help practitioners putting an end to erroneous intuitions. This suggests that it is unfortunate for practitioners to rely on intuition as much as they do on analysis.

Kahneman and Frederick, the "ball and bat" puzzle, and the famous "Linda" experiment further shows the untrustworthiness of intuition, and show how people rely on representativeness, overruling basic rules of logics. In "Linda," respondents were given a description of a woman who was very representative of a feminist. This made the respondents claim that it was more probable that she was a bank teller *and* a feminist than that she was only a bank teller. One can easily imagine a similar fallacy when software practitioners choose between for example software process models: If process model X seems more like a "typical" process model than process model Y does, people will substitute process model X's representativeness for suitability — instinctively disregarding rational analysis, just like with "Linda". Practitioners are unlikely to be self-aware of this, and the number of practitioners who in practice rely on intuition instead of analysis could be higher than the self-reported figure.

As noted in section 3.3.2, Griffin and Tversky claim that when "predictability is very low, [. . .] experts may be more prone to overconfidence than novices." The faith practitioners have in experts is thus not necessarily always a good thing. But listening to experts is of course valuable in many situations. The fact that experts systems — software that "stores" expertise in an attempt to replicate experts' conduct — is a major field in computer science should be enough as evidence to support the value of experts' advice. Also, researchers, experts, and practitioners have common goals, and all parties should gain from closer collaboration. Such collaboration has been described by Jarvis, who use the expression "practitioner–researcher" to describe practitioners who also do research, typically "expert practitioners working toward graduate degrees as part-time students" [23, p. 5]. Cooperation between industry and research is also increasingly a priority for funding bodies.

Attitudes to and appliance of research 53.7% of the respondents claimed to apply results from research in software engineering in their work (Question 7, section 5.3.7). 23.9% claimed not no, while 22.1% did not know.

The mean confidence in research in software engineering was $m = 4.10$ (on a scale where 1 was "nothing" and 7 "very much"). On Question 5, where the respondents were asked how much they kept up with research, the mean value was 3.85 (1 = "nothing", 7 = "very much").

The exploratory analysis showed little or no correlation between the age, education and industry experience of the respondents and their confidence in research or willingness to apply research in their work. There is a connection between having confidence in research and applying it: The partitioning example shown in section 5.4.3 shows that among the group which have a confidence in research ≥ 4, 79.4% claim to apply research in their work, while in the group that has a confidence < 4, only 44.0% apply research. The latter connection is hardly surprising, but we found it more striking that the education level did not affect the confidence in or appliance of research – we would have guessed that practitioners with advanced degrees (i.e. a master or higher) would be more positive towards research than those without advanced degrees.

Software practitioners' attitudes could be compared to medical practitioners' attitudes, as discussed in section 2.4. On a scale from 0 ("strongly disagree") to 100 ("strongly disagree"), the median score that medical practitioners gave to the statement "evidence–based medicine improves patient care" was 70. This could be interpreted as a measure of how confident medical practitioners are in research, and their level of confidence is comparable to (but somewhat higher than) the confidence SE practitioners have in research ($M = 4$ on a scale from 1 to 7).

Osterweil et al. (as discussed in section 3.2) claimed that “lasting impact [of research] seems to come most readily from ongoing interactions between research and practice.” If it is correct that there is such a high number of practitioners who use research in their work, this is joyful news to both researchers and practitioners alike, as both gain from ongoing interactions. Osterweil et al. found that it normally takes a long time (typically 10 to 20 years) for new ideas to move from research into widespread practice. This initially made us speculate about practitioners perceiving research as “irrelevant” or even “outdated”; but our data analysis does not support these speculations.

6.1.3 Obstacles to use research

SRQ3 Which obstacles exist for software practitioners to use research?

This SRQ was covered by Question 8. The barriers are repeated (in descending order) below for the sake of convenience, and are also shown in figure 5.8.

Barrier	*N*
Too busy meeting immediate goals or deadlines	62
Lack of personal time	56
Research results are hard to find	36
Research is not relevant to my needs	26
Attitudes or expectations of customers	20
My organization does not encourage the use of research	13
Attitudes of colleagues	9

As reported in section 4.3.3, there was a total of 113 completed responses, of which 103 had one or more items ticked at Question 8.

The most widely reported barrier was “too busy meeting immediate goals or deadlines” (62/113), while “lack of personal time” came second (56/113). Attitudes of both customers and colleagues as well as discouraging organizations were not seen as barriers by many respondents. This is in line with medical practitioners, who had “lack of personal time” as their by far most widely reported barrier (see section 2.4)

We consider the barriers reported as encouraging to researchers. Prejudices against research are not at all widespread in organizations and among customers, and relatively few practitioners see research as irrelevant to their needs. The most widely reported barriers — both variants of the dreaded “time squeeze” — are outside researchers’ control. Eliminating these barriers is a managerial issue, if companies wish to encourage wider deployment of research results in their organizations.

6.1.4 Topics researchers should focus on

SRQ4 Which topics do software practitioners feel that researchers should focus on?

The questionnaire elicited a number of topics that respondents felt that researchers should focus more on. We then classified these topics into exactly one Software Engineering category using ACM's *Computer Classification System* (CCS), and removed all topics that could not be fit into any of these. 73 topics were classifiable, while 60 were dropped as too general, ambiguous, or frivolous.

Most desired topics The following table shows the topics our respondents felt that researchers should focus on. All numbers are in percent of all applicable responses. The raw count can be found in section 5.3.9.

Category	**Percantage**
D.2.1 Requirements/Specifications	5.48
D.2.2 Design Tools and Techniques	1.37
D.2.3 Coding Tools and Techniques	15.07
D.2.4 Software/Program Verification	8.22
D.2.5 Testing and Debugging	6.85
D.2.6 Programming Environments	2.74
D.2.7 Distribution, Maintenance, and Enhancement	9.59
D.2.8 Metrics	1.37
D.2.9 Management	34.25
D.2.10 Design	1.37
D.2.11 Software Architecture	6.85
D.2.12 Interoperability	6.85
D.2.13 Reusable Software	0

The clear "winner" is "Management", with 34.25%. Other topics of importance to practitioners are "Coding Tools and Techniques" (15.07%), and "Distribution, Maintenance, and Enhancement" (9.59%). The least mentioned topics were "Reusable Software" (0%), "Metrics" (1.37%), "Design Tools and Techniques" (1.37%) and "Programming Environments" (2.74%).

The "Management" section of the CCS consists of these sub-items:

- Copyrights
- Cost estimation
- Life cycle
- Productivity
- Programming teams
- Software configuration management
- Software process models (e.g., CMM, ISO, PSP)
- Software quality assurance (SQA)
- Time estimation

We have not done a complete and formal classification of the "Management" answers into these sub-topics, but it is our clear impression that "Software process models" is the most desired sub-topic. 5 respondents mentioned Agile methods, clearly a topic of interest for practitioners.

One should keep in mind that the total number of topics given in response to this question was not that high (N = 133). One should thus not emphasize the internal order of the least mentioned topics, as only a couple of respondents mentioned these.

A lot of the topics were put into the "N/A" class because they were not included in the Software Engineering part of the CCS. The most obvious examples were concurrency and parallelism, which were mentioned by many but are classified by the CCS under "Operating Systems". Our scope is software engineering, and the internal concepts of Operating Systems are outside that particular academic discipline. We encourage researchers in other, related disciplines to conduct similar surveys to get feedback on practitioners' wishes when it comes to the topics of research in the respective disciplines.

The many "not applicable" topics we collected could be caused by a much narrower definition of "Software Engineering" used by us (as researchers) than what is commonly thought of in industry. This discrepancy could also be caused by confusing terminology in Norwegian, where the disciplines that in English are known as "Computer Science" and "Software Engineering" normally are lumped together as "Informatikk" (English: *Informatics*). When we ask for topics in Software Engineering, respondents could have a hard time differentiating between the different parts of "Informatics".

Desired topics versus what researchers actually focus on The following table shows a comparison of the desired topics and the topics of published research papers. All numbers are percentages.

Category	Cai&Card	Desired topics
D.2.1 Requirements/Specifications	5.91	5.48
D.2.2 Design Tools and Techniques	9.66	1.37
D.2.3 Coding Tools and Techniques	4.77	15.07
D.2.4 Software/Program Verification	17.27	8.22
D.2.5 Testing and Debugging	19.2	6.85
D.2.6 Programming Environments	4.55	2.74
D.2.7 Distribution, Maintenance, and Enhancement	5.8	9.59
D.2.8 Metrics	8.3	1.37
D.2.9 Management	11.02	34.25
D.2.10 Design	1.36	1.37
D.2.11 Software Architecture	7.73	6.85
D.2.12 Interoperability	1.93	6.85
D.2.13 Reusable Software	2.50	0

Figure 6.1 shows the same data graphically. The abbreviations used in the figure should be self-explanatory. The first column refers to CCS's D.2.1 for actually researched topics, the second column to D.2.1 for desired topics, the third column to

D.2.2 for actual topics, the fourth to D.2.2 for desired topics, and so on.

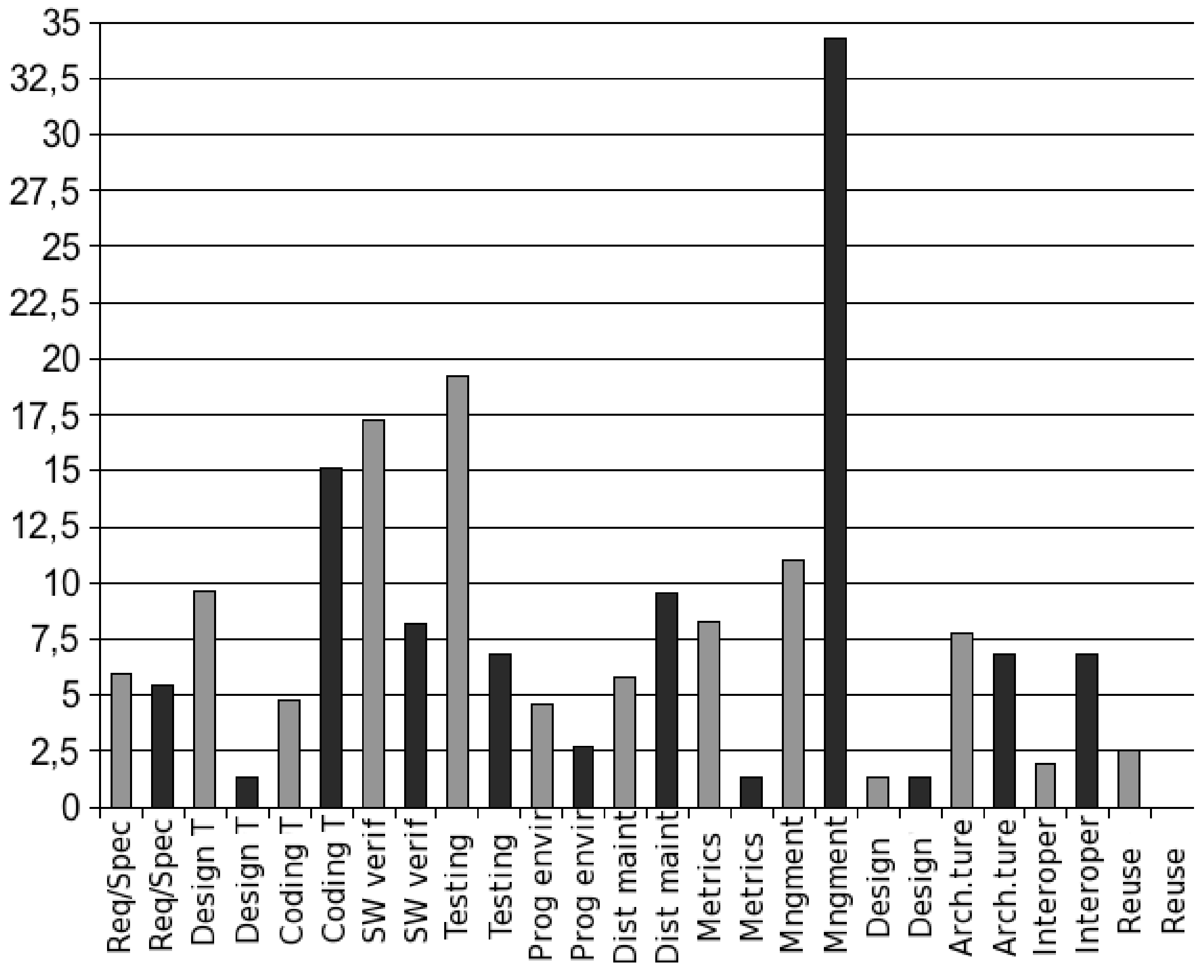


Figure 6.1: Researched topics (light gray) and topics desired by practitioners (dark gray).

The most apparent under-researched topic is "Management". 11.02% of the research papers examined by Cai and Card concerned this general topic, while 34.25% of the (classifiable) topics mentioned by our respondents concerned it. "Coding Tools and Techniques" is similarly under-researched.

Among the over-researched topic, the most prevalent are "Design Tools and Techniques", "Software / Program Verification", and "Metrics".

6.2 Threats to validity

According to Jedlitschka et al., "all threats that might have an impact on the validity of the results need to be discussed" when reporting experiments in Software Engineering [24]. These include (1) *threats to construct validity*, (2) *threats to internal validity*, (3) *threats to external validity*, and if applicable, (4) *threats to conclusion validity*.

Even though a survey differs from an experiment in many important ways, we still use these four groups of validity threats as our starting points when discussing the validity of our survey. These four types of validity threats are discussed in the sections below.

6.2.1 Construct validity

Construct validity concerns whether the scales used in a study actually represents the constructs in the real world, and include both design threats and social threats.

Fink and Litwin describe construct validity as "the most valuable and yet the most difficult way of assessing a survey instrument" [11, p. 41]. They note that "[i]t is difficult to understand, to measure, and to report", and that "[t]his form of validity is often determined only after years of experience with a survey instrument".

The construct validity threats discussed below are adapted from Wohlin et al. [52, pp. 71–72].

Hypothesis guessing Some practitioners taking part in the survey may try to find out the purpose or intended result of the study, and adjust their behavior, either positively or negatively, according to their attitudes. For example, noting that the survey originated from a scholarly institution, some participants might want to please the surveyors by stating that they value research more than experts' advice. We found no way of preventing this possible effect without sacrifying some of our ethical standards, as discussed in section 6.3.

Researcher expectancies Researchers can bias the results of a survey (both consciously and unconsciously) based upon the results they expect. We have countered this validity threat by involving different people with different expectations in the survey.

Inadequate pre-operational explication of constructs This concerns if the constructs are sufficiently defined before being translated into measures. The two pilot iterations (where we asked the pilot subjects about what terms like for example "scientific journal" and "expert" meant) made us reasonably sure that the constructs were both clearly defined and consistently understood.

A related threat arises from the erroneous formulation discussed in section 4.2.4. However, the fact that nobody bothered to notify us at the javaBin event suggests that this error either went unnoticed by respondents or that they clearly understood what we meant. We see no apparent way this error can introduce an extra bias towards either intuition or analysis.

6.2.2 Internal validity

Wallen and Fraenkel claim that the four main threats to internal validity in survey research are “mortality, location, instrumentation, and instrument decay” [51, p. 390].

Mortality is not a concern to us, as this threat only is applicable in longitudinal studies (i.e. studies where one observe the same subjects over a long period of time). A location threat arrises when the data collection is carried out in places that may affect responses — Wallen and Fraenkel use the example of a survey concerning attitudes to the police carried out in a police station. We cannot see that this applies to our survey.

Instrument decay is for surveys usually limited to interview surveys, where the interviewer could get tired or be in a rush. This will of course not apply to a web form. Instrument defects (i.e. bugs in *SurveyMonkey*'s software) cannot be completely ruled out, but this is unlikely, given the system's wide deployment.

6.2.3 External validity

King and He give the following definition of external validity [28]:

> External validity refers to the generalizability of sample results to the population of interest, across different measures, persons, settings, or times. External validity is important to demonstrate that research results are applicable in natural settings, as contrasted with classroom, laboratory, or survey-response settings.

Coverage error Coverage error is introduced when the frame from which the sample is drawn does not include all of the relevant characteristics in the population to which inferences are to be drawn. According to King and He [28], coverage error affects the external validity of the results of studies

- that rely on self-reporting by individual respondents, whether the measures being reported are factual (e.g. age) or perceptual (e.g. responses on a Likert scale), and
- in which the population to which inferences are to be drawn may be individuals operating in the “natural” world (e.g. managers or IT professionals) or organizations for whom individual responses are taken to, or aggregated to, be representative.

We believe that javaBin and companies Y and Z are highly representative of the ICT industry. JavaBin's membership base consists of practitioners from all kinds of Norwegian ICT companies. Java technologies can be used in most phases of the software development process, and javaBin comprises practitioners in a wide variety of roles. We see no apparent reason why the population of Java practitioners should be inherently different from for example the population of *.NET* practitioners or other subpopulations of Norwegian ICT practitioners. The same applies to companies Y and Z, which we consider typical of Norwegian software houses with 100+ developers. However, our sample was drawn from a young (age $<$ 40 and industry experience $<$ 15 years) segment of the population, and we should ideally have included some more of older and more experienced practitioners as well.

Access to computers, e-mail, and the World Wide Web is not evenly distributed in a society, and it is reasonable to believe that richer persons have better access than poorer people. This may in general lead to a coverage bias when using on-line surveys, as the wealthy segment that has access to computers will be over-sampled. We believe that this is not a problem among software professionals, and assume that all or nearly all in the software industry have access to the World Wide Web. We are reasonably sure that using this medium does not lead to any undue coverage error or response bias. One could have offered the respondents the option of receiving the questionnaire by regular mail just in case. However, Wade and Parent conducted an on-line survey among ICT professionals in 2001 offering a regular mail option, and noted that no respondents took advantage of this option [50]. With Internet coverage being much more widespread now than eight years ago, we believe that such an offer would be completely unnecessary.

Nonresponse error King and He remark that nonresponse error is introduced [28]

> if non-respondents are different from respondents in terms of characteristics that are relevant to the study [Dillman, 2000]. In general, error due to nonresponses is presumed to be in direct relationship to increases in the rate of nonresponse and the level of variation in the true attitudes, beliefs or perceptions that are being assessed [American Association for Public Opinion Research, 2004].

King and He further classify nonrespondents as "active" and "passive": "Active (purposeful) non-respondents may decide that completing a survey is too time-consuming, or irrelevant to their job or organization or just that 'I get too many surveys."', while the passive nonrespondents "intend to respond, but forget or 'just didn't get to it."' [28].

As discussed in section 4.3.2, 113 of the 145 respondents who started the survey completed it. Respondents were counted as "completed" if they pressed the "submit" button at the end of the questionnaire, and our statistical analysis was done on the set of 113 completed responses.

Describing non-repondents (i.e. those who did not complete the survey or did not answer at all) is hard. There are several formal approaches that can be taken, such as surveying a number of non-respondents by phone, or comparing late and early respondents using the assumption that late respondents are more like non-respondents than early responders are. We chose a simplified implementation of the latter method, by calculating and comparing the means of questions 3 and 4 from the 20 first and 20 last on-line respondents using the javaBin collector. These questions and the number of respondents were chosen arbitrarily.

For Question 3, $m_{20first} = 3.71$ and $s_{20first} = 1.25$, while $m_{20last} = 4.00$ and $s_{20last} = 1.00$. The mean m of all responses was 3.92, with $s = 1.16$, U95 = 4.14 and L95 = 3.70.

For Question 4, $m_{20first} = 4.60$ and $s_{20first} = 1.24$, while $m_{20last} = 4.94$ and $s_{20last} = 1.06$. m was 4.51, $s = 1.14$, U95 = 4.76 and L95 = 4.33.

For both questions 3 and 4, the means of the 20 last javaBin respondents were higher than the means of the 20 first javaBin respondents. If one uses the assumption that late responders are more like nonresponders than early responders are, it could be that the reported values are underreported. It could have been interesting to look further into this issue, but time constraints forced us to focus on other things.

6.2.4 Conclusion validity

(Statistical) conclusion validity refers to whether the conclusions reached in a study are correct. This is directly related to statistical hypothesis testing, and the possibility of type I and type II errors (i.e. rejecting the null hypothesis when the null hypothesis is true, and failing to reject the null hypothesis when the null hypothesis is false).

We did not do statistical hypothesis testing on our data, as we rather chose the approach known as exploratory data analysis. Exploratory data analysis eliminates the possibility of type I and type II errors, but one should nevertheless discuss the suitability of our statistical methods. The perhaps most controversial design choice we made was using means instead of medians as a measure of the distribution centers. A median is often a better measure of strongly skewed distributions, but using medians has the clear disadvantage of giving us considerably weaker statistical tools at our disposal. Both means and medians are reported in section 5.3. The difference between these two measures was small for most variables, and we are thus confident that that using means is good enough for our purpose.

6.3 Research ethics

According to Vinston and Singer, the Empirical Software Engineering community has yet to develop its own code of research ethics [49]. They thus recommend ESE researchers to apply codes from related disciplines.

We have used guidelines compiled by Sarantakos when conducting our study [41, pp. 18-24]. These guidelines, aimed at social researchers, list these seven basic principles of ethical social research as the most important:

- Proper identification; not giving the respondents false impressions of the researcher or the sponsor.

- Clear information as to the type of questions, the degree of question sensitivity or stress and the possible (true) consequences of the questioning and the research in general.
- Concern with the welfare of the respondents, including having regard for mental and physical health and safety, embarrassment, guilt, discomfort, hazards or risks to the respondents [Bailey, 1982, 1988; Sproull, 1988; Vlahos, 1984].
- Free and informed consent; for example, not putting pressure on or deceiving the respondents.
- Right of privacy regarding their private life, sensitive issues or answering questions they dislike.
- The right to anonymity, meaning that the respondents' contributions must remain anonymous.
- The right to confidentiality; the respondents' contributions should not be made available to other people.

We can not see any major problems with our our study in regard to these principles. The most relevant principles are the last two (anonymity and confidentiality), in addition to the "free and informed consent" principle.

Consent We did not include explicit consent forms, but believe that the implicit consent given when the respondents choose to actively answer the online questionnaire is sufficient.

The deployments in companies Y and Z were done in agreement with the respective managements. We promised these access to the data we collected, and they thus had some interest in getting as many of their employees as possible to participate. It is therefore possible that there has been some (formal or informal) pressure on their employees to participate, but the low response rates suggest that there has not.

Anonymity Our questionnaire said that the survey would be anonymous, and we did not record the respondents' names. It was voluntary to state one's email address.

It is technically possible for the survey administrators (i.e. ourselves and also possibly members of the *SurveyMonkey* staff) to associate the respondents' e-mail addresses with their answers, but we have no reason to believe that this has been done.

Confidentiality We ensured confidentiality by not associating names, e-mail addresses or IP addresses with data that is publicly available. We cannot see any way that data can be linked to individual respondents.

6.4 Summary

This chapter has summarized and discussed the implications of the results of our survey.

SRQ1 is discussed in section 6.1.1. Software practitioners mostly learn about new techniques and technologies from experts and friends. Sources such as web forums, mailing lists, blogs, and other social media are also important. Scientific conferences and scientific journals are the least used sources of information, but practitioners who read scientific journals have more confidence in research than practitioners who do not. There is a high correlation between those who seek the advice of experts and friends when making decisions and those who rely on experts and fiends when learning about new technologies.

In section 6.1.2, we discussed SRQ2. Experts are valued more than research when practitioners are considering new technologies. On average, practitioners do not differentiate much between independent researchers and industrial researchers. Practitioners give about equal weight to their own analysis, the advice of experts, and intuition when making important decisions. Those who report that the attitudes of colleagues crate a barrier to applying research are more likely to use gut feeling over analysis.

A majority of practitioners claimed to apply research from software engineering in their work, and there is unsurprisingly a connection between having confidence in research and applying it. However, we did not find a connection between neither the education, age, nor industry experience of practitioners and their attitudes to or appliance of research.

Section 6.1.3 described the obstacles to using research. "Too busy meeting immediate goals or deadlines" and "lack of personal time" were the most widely reported barriers, while neither the attitudes of customer or colleagues, nor organizational discouragement were seen as barriers. This should be encouraging to researchers.

As discussed in section 6.1.4, the most widely reported research topics of interest to practitioners were related to management and to coding tools and techniques. However, many of the topics we collected were unclassifiable using ACM's software engineering categories, something which might have been caused by different terminology in English and Norwegian.

In comparison to the list of topics from recent papers in research in software engineering (compiled by Cai and Card and discussed in section 3.1), topics related to management and to coding tools and techniques are under-researched. Design tools and techniques, software / program verification, and metrics are over-researched.

We discussed threats to validity in section 6.2. Threats to construct validity are hard to get rid of, and were discussed in section 6.2.1. We hope that our pilot testing made the constructs clearly defined and consistently understood. Threats to internal validity (section 6.2.2) are not that severe to us, but the coverage bias discussed under external validity threats (section 6.2.3) is more important. We believe that companies Y and Z and javaBin are highly representative of the Norwegian ICT industry, even though the sampled population was somewhat young. In section 6.2.4, we discussed why we used means instead of medians when doing statistical analysis.

Section 6.3 concerned research ethics, and we discussed how we took care of the respondents' anonymity and how we ensured confidentiality.

Chapter 7

Conclusion

7.1 Summary

This thesis has described a survey of software practitioners which the purpose of eliciting their attitudes to research. The study was motivated by the need for researchers in empirical software engineering to better understand practitioners.

Researchers have proposed an approach to software engineering practice called evidence-based software engineering (EBSE). Applying EBSE involves searching (scientific) literature, critically appraising the evidence, and integrating the appraised evidence with practical experience and customers' values to make decisions. However, if practitioners do not find research valuable, relevant, or trustworthy, they will certainly not approve of EBSE.

To elicit these opinions, we deployed a questionnaire in two Norwegian software houses and in one Norwegian Java users' group. There were 113 respondents, who answered questions about how they make technology decisions, which sources they use for gaining knowledge about new technologies, and how they rate research / analysis in comparison to intuition and experts' advice. They also answered questions regarding how much they keep up with research, whether they apply it, whether they have confidence in research, and whether barriers prevent them from applying research. We also asked the respondents to name topics that they feel software engineering researchers should focus on more.

We analyzed the data using JMP, a software package offering a comprehensive set of statistical tools. Utilizing JMP, we applied a statistical technique called recursive partitioning on the data to explore relationships between variables.

7.2 Contributions

Software practitioners overwhelmingly rely on colleagues and friends when learning about and considering implementing new technologies. The advice of experts is also important. Information form web sites, web forums, social media, and ICT newspapers are also important sources of knowledge, along with industrial conferences.

Most practitioners neither consult nor rely on researchers when looking for knowledge about new technologies, but those who do tend to have more confidence in research. Most practitioners do not differentiate between industrial and independent (academic) researchers when assessing information. Practitioners rely almost equally on analysis, experts' advice and intuition when making important decisions.

A majority of practitioners claim to apply research in software engineering in their work. Being too busy meeting immediate goals / deadlines and lack of personal time were the two most widely reported barriers to applying research, while neither organizational inertia, attitudes of colleagues or customers, nor the relevance of research were seen as barriers. The reported confidence in research in software engineering was moderate, and not fundamentally different from how medical practitioners value research. Unsurprisingly, practitioners who have more confidence in research are also more likely apply it in their work.

Topics related to management were mentioned most often when practitioners were asked which topics researchers should focus more on. Topics related to coding tools and techniques were also important. In comparison to the topics in demand from practitioners, researchers focus too much on design tools and techniques, software/program verification, and metrics.

Neither the age, the education, nor the industry experience of practitioners significantly influenced their attitudes to research.

7.3 Future Work

Refining the survey and deploying it more widely could produce more interesting findings. In particular, we would like to have it specifically deployed among managers, as very few of the respondents reported working in management. Eliciting differences in managers and lower level practitioners' attitudes to research could be useful, as managers are more likely to make important decisions than practitioners. We also suggest doing more formal rounds of pilot testing, so that one can be more sure that the constructs are consistently defined and understood. Deploying the survey internationally would also be interesting, as the findings may be more generalizable in an international setting.

An interesting supplement to the survey would be Repertory Grid Analysis. This is a semi–structured interview technique, where the interviewee himself determines the most important concepts of the subject matter of the interview, and then ranks these concepts according to his own value system. See for example Rognerud and Hannay for more information [39]. We initially planned doing a round of such interviews, but had to focus on our survey because of a tight schedule.

Bibliography

[1] A. L. Abel, N. B. Sardone, and S. Brock. Simulation in the college classroom: Enhancing the survey research methods learning process. *Information Technology Learning and Performance Jornal*, 23(2):39–46, 2005.

[2] How to Use the Computing Classification System : How to classify works using ACM's Computing Classification System. ACM website. [online] `http://www.acm.org/about/class/how-to-use`. Archived at `http://www.webcitation.org/5ePp3xMQI`. Accessed: 2009-02-05.

[3] Frederick P Brooks jr. *The Mythical Man–Month : Essays on Software Engineering Anniversary Edition*. Addison–Wesley, 1995.

[4] Kai-Yuan Cai and David Card. An analysis of research topics in software engineering – 2006. *Journal of Systems and Software*, 81(6):1051–1058, June 2006.

[5] Brad J. Cox. There is a silver bullet. In Nick Heap, editor, *Information technology and society*, pages 377–386. SAGE, 1995.

[6] Jenny A. Darby. Open-ended course evaluations: a response rate problem? *Journal of European Industrial Training*, 31(5):402–412, 2007.

[7] George E. M. Ditsa. Combating the software crisis: Service quality and force–field approach. In Mehdi Khosrowpour, editor, *Effective Utilization and Management of Emerging Information Technologies*, pages 752–773. Idea Group, 1998.

[8] T. Dybå, B.A. Kitchenham, and M. Jørgensen. Evidence-based software engineering for practitioners. *Software, IEEE*, 22(1):58–65, Jan.-Feb. 2005.

[9] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833 – 859, 2008.

[10] H. Erdogmus, M. Morisio, and M. Torchiano. On the effectiveness of the test-first approach to programming. *Software Engineering, IEEE Transactions on*, 31(3):226–237, March 2005.

[11] Arlene Fink and Mark S. Litwin. *The Survey Kit: How to assess and interpret survey psychometrics*. SAGE, second edition, 2002.

[12] Steven E. Finkel, Thomas M. Guterbock, and Marian J. Borg. Race of interviewers effects in a prediction poll virginia 1989. *Public Opinion Quarterly*, 55:313–330, 1991.

[13] Floyd J. Fowler Jr. *Improving Survey Questions*. Applied Social Research Method Series, Volume 38. Sage Publications, 1995.

[14] Floyd J. Fowler Jr. *Survey Research Methods*. Applied Social Research Method Series, Volume 1. Sage Publications, 2002.

[15] Steven Fraser, Dave Astels, Kent Beck, Barry Boehm, John McGregor, James Newkirk, and Charlie Poole. Discipline and practices of tdd: (test driven development). In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 268–270, New York, NY, USA, 2003. ACM.

[16] Pearl Friedman. A second experiment on interviewer bias. *Sociometry*, 5(4):378–381, 1942.

[17] Robert L. Glass. The standish report: does it really describe a software crisis? *Commun. ACM*, 49(8):15–16, 2006.

[18] Alan Gordon. Surveymonkey.com–web-based survey and evaluation system: http://www.surveymonkey.com. *The Internet and Higher Education*, 5(1):83 – 87, 2002.

[19] Dale Griffi and Amos Tversky. The weighing of evidence and the determinants of confidence. *Cognitive Psychology*, 4(3):411–35, 1992.

[20] Jo E. Hannay, Erik Arisholm, Harald Engvik, and Dag I.K. Sjøberg. Effects of personality on pair programming. *IEEE Transactions on Software Engineering*, 12 Jun. 2009. IEEE computer Society Digital Library. IEEE Computer Society, <http://doi.ieeecomputersociety.org/10.1109/TSE.2009.41>.

[21] About IEEE Software. IEEE website. [online] `http://computer.org/portal/site/software/menuitem.538c87f5131e26244955a4108bcd45f3/index.jsp?&pName=software_level1&path=software/content&file=about.xml&xsl=article.xsl&`. Archived at `http://www.webcitation.org/5gaxurlFa`. Accessed: 2009-05-07.

[22] Ulf Jakobsson. Statistical presentation and analysis of ordinal data in nursing research. *Scandinavian Journal of Caring Sciences*, 18(4):437–440, 2004.

[23] Peter Jarvis. *The Practitioner-Researcher: Developing Theory from Practice*. Jossey-Bass, 1998.

[24] Andreas Jedlitschka, Marcus Ciolkowski, and Dietmar Pfahl. Reporting experiments in software engineering. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 201–228. Springer, 2008.

[25] David Richard Johnson and James C. Creech. Ordinal measures in multiple indicator models: A simulation study of categorization error. *American Sociological Review*, 48(3):398–407, 1983.

[26] Magne Jørgensen and Kjetil Moløkken-Østvold. How large are software cost overruns? a review of the 1994 chaos report. *Information and Software Technology*, 48(4):297–301, 2006.

[27] Daniel Kahneman and Shane Frederick. A model of heuristic judgment. In Keith J. Holyoak and Robert G. Morrison, editors, *The Cambridge Handbook of Thinking and Reasoning*, pages 267–294. Cambridge University Press, 2005.

[28] William R. King and Jun He. External validity in is survey research. *Communications of the Association for Information Systems*, 16:880–894, 2005.

[29] B.A. Kitchenham, T. Dybå, and M. Jørgensen. Evidence-based software engineering. *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 273–281, 23-28 May 2004.

[30] Barbara A. Kitchenham and Shari L. Pfleeger. Personal opinion surveys. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 63–92. Springer, 2008.

[31] Edith D. de Leeuw and Johannes van der Zouwen. Data quality in telephone and face to face surveys: A comparative meta–analysis. In Robert M. Groves, Paul P. Biemer, and Lars E. Lyberg, editors, *Telephone Survey Methodology*, pages 283–300. John Wiley and Sons, 2001.

[32] Chris Mann and Fiona Stewart. *Internet Communication and Qualitative Research*. SAGE, 2000.

[33] Alastair McColl, Helen Smith, Peter White, and Jenny Field. General practitioners' perceptions of the route to evidence based medicine: a questionnaire survey. *BMJ*, 316:361–365, January 1998.

[34] Mary A. Meyer and Jane M. Booker. *Eliciting and analyzing expert judgment: a practical guide*. Society for Industrial and Applied Mathematics, 2001.

[35] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W. H. Freeman and Co, fifth edition, 2006.

[36] Leon J. Osterweil, Carlo Ghezzi, Jeff Kramer, and Alexander L. Wolf. Determining the impact of software engineering research on practice. *Computer*, 41(3):39–49, 2008.

[37] Dewayne E. Perry, Adam A. Porter, and Lawrence G. Votta. Empirical studies of software engineering: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 345–355, New York, NY, USA, 2000. ACM.

[38] Samuel T. Redwine, Jr. and William E. Riddle. Software technology maturation. In *ICSE '85: Proceedings of the 8th international conference on Software engineering*, pages 189–200, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press.

[39] Heidi Jacobsen Rognerud and Jo Erskine Hannay. Challenges in enterprise software integration: An industrial study using repertory grids. In *Empirical Software Engineering and Measurement (ESEM)*, 2009.

[40] John Sall, Lee Creighton, and Ann Lehman. *JMP Start Statistics*. SAS Institute, fourth edition, 2007.

[41] Sotirios Sarantakos. *Social Research*. Palgrave Macmillian, third edition, 2005.

[42] SAS Institute. *JMP Statistics and Graphics Guide, Release 7*, May 2007.

[43] Janice Singer, Susan E. Sim, and Timothy C. Lethbridge. Software enigneering data collection for field studies. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 9–34. Springer, 2008.

[44] Dag I. K. Sjøberg, Tore Dybå, and Magne Jørgensen. The future of empirical methods in software engineering research. In *FOSE '07: 2007 Future of Software Engineering*, pages 358–378, Washington, DC, USA, 2007. IEEE Computer Society.

[45] Ian Sommerville. *Software Engineering*. Addison–Wesley, 8th edition, 2007.

[46] Frank Stanton and Kenneth H. Baker. Interviewer-bias and the recall of incompletely learned materials. *Sociometry*, 5(2):123–134, 1942.

[47] A Tversky and D. Kahneman. Extensional vs. intuitive reasoning. In D. Kahneman, P. Slovic, and A. Tversky, editors, *Judgment under uncertainty: Heuretics and biases*, pages 84–98. Cambridge University Press, 1982.

[48] Amos Tversky and Daniel Kahneman. Belief in the law of small numbers. *Psychological Bulletin*, 76(2):105–110, 1971.

[49] Norman G. Vinston and Janice Singer. A practice guide to ethical research involving humans. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 229–256. Springer, 2008.

[50] M. R. Wade and M. Parent. Relationships between job skills and performance: A study of webmasters. *Journal of Management Information Systems*, 18(3):71–96, 2001.

[51] Norman E. Wallen and Jack R. Fraenkel. *Educational research: a guide to the process*. Lawrence Erlbaum Associates, second edition, 2001.

[52] Claes Wohlin, Per Runeson, Martin Høst, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Springer, 1999.

[53] Bruno D. Zumbo and Donald W. Zimmerman. Is the selection of statistical methods governed by level of measurement? *Canadian Psychology*, 34(4):390–400, 1993.

Appendix A

List of survey questions

This appendix shows the list of questions from the questionnaire discussed in chapter 4, section 4.3.

The questionnaire has been reproduced as faithful as possible. This means that the wording is reproduced verbatim, and that the typography is preserved as much as possible. There are however some typographic changes that are unavoidable when converting it from a HTML format into LaTeX.

A.1 #1. Intro

This survey is a part of a research project at the software engineering department at Simula Research Laboratory in Oslo, Norway. It is aimed at software professionals, both practitioners and managers / executives.

Participating will take approximately 10 minutes, and your participation is greatly appreciated. One of the participants will win an Apple iPod nano 8GB.

1. If you would like a chance to win an iPod nano, please enter your email address below. Your email address will not be associated with your responses to the questions.

- (Text field)

A.2 #2. Part 1/2: Introductory questions

1. Age How old are you?

- Drop down box : "19 or less" → "71 or more"

2. In which country are you currently working?

- (Text field)

3. What is your current position? (choose one)

- Developer
- Product Manager
- QA Manager
- Other (please specify)

4. Industry experience How many years of industry experience do you have?

- Drop down box : "0 → 4" → "40 or more"

5. What is your highest level of formal education? (choose one)

- High school (Videregående skole) or less
- Some college, but no degree
- A bachelors degree or equivalent (Høyskoleingeniør, cand.mag. etc.)
- A masters degree or equivalent (Civil engineer, cand.real. etc.)
- A doctoral degree (ph.d, dr.ing, dr.scient etc.)

A.3 #3. Part 2/2

All scales in this section are ordinal. This means that the elements have a simple order, so that for example 4 is better than 3, and that 3 is better than 2. However, it does not mean that 4 is twice as good as 2.

1. You are considering a new technique or technology in IT development. You have only superficial prior knowledge about this technique or technology.

On a scale where 1 is "not at all" and 7 is "very much", how important for you is advice from the following persons and entities when you are making your decision?

- A colleague or friend with experience with this technique

- An external expert / guru
- Independent researchers (MIT, The Norwegian University of Science and Technology)
- Industrial researchers (IBM, Sun) and commercial research companies (The Standish Group)
- Information from the vendor or supplier (success stories, product demonstrations)
- Other (please specify)

2. When looking for knowledge about new development techniques and technologies, to what extent do you consult the following sources?

- Experts/gurus, colleagues, and friends in the IT industry
- Industrial conferences (JavaZone etc.)
- Popular scientific journals and magazines
- Scientific conferences
- Scientific journals (IEEE Transactions on Software Engineering etc.)
- Web forums, mailing lists, blogs and other social media
- Websites and IT newspapers (Slashdot, Computerworld)
- Other sources (please specify)

3. It has been decided that your organization needs to radically renew its software development process model. You are responsible for evaluating the alternatives, and there are two obvious alternatives, X and Y. You have assessed the available literature, and while there are pros and cons to both alternatives, your analysis of the literature shows that alternative X is probably the best choice for your organization.

Nevertheless, you have a strong feeling (gut feeling, "magefølelse") that alternative Y is best suited for your organization, even though you find it hard to explain why.

On a scale where 1 is "gut feeling only" and 7 is "analysis only", how much weight would you give to your analytical findings versus your gut feeling?

- (choose one, leave blank if you do not know)

4. Your organization needs a new testing framework. You have found reliable research claiming that using framework F will most of the time find more bugs and increased productivity.

You have also consulted an expert whom you trust from previous experience. He claims that framework F is worthless, and strongly recommends using framework G instead.

On a scale from 1 ("research only") to 7 ("expert advice only"), how would you give weight to the two recommendations?

- (choose one, leave blank if you do not know)

5. On an scale from 1 ("not at all") to 7 ("very much"), how much do you keep up with research in software engineering?

- (choose one, leave blank if you do not know)

6. On an scale from 1 ("nothing") to 7 ("very much"), how much confidence do you have in research in software engineering?

- (choose one, leave blank if you do not know or do not keep up with research)

7. Do you apply results from research in software engineering in your work?

- Yes
- No
- Don't know

8. Do one or more of these barriers prevent you from applying research in your work? (choose all that apply, choose none if no alternative applies)

- My organization does not encourage the use of research
- Too busy meeting immediate goals or deadlines
- Research results are hard to find
- Lack of personal time
- Research is not relevant to my needs
- Attitudes of colleagues

- Attitudes or expectations of customers
- Other (please specify)

9. Name up to three topics that you feel software engineering researchers should focus on more.

- (Text field)
- (Text field)
- (Text field)

10. Which software engineering activity or job task is your main daily focus at present?

- (Text field)

Appendix B

List of topics from Question 9

This appendix shows the topics we collected in Question 9 and our categorization of them. See section 5.3.9 for details.

All responses are reproduced verbatim in the column to the left. The corresponding CCS categories, assigned by us, are in the center column. The column to the right shows the respondents' responses to Question 10, i.e. their main focus at present.

Response	**Category**	**Current activity**
Usability of software	N / A	Developing web applications
Time-saving (by software developing)	N / A	Testing
Time to market with articles, they are often dated when they are published	N / A	Development
Successful vs unsuccesful projects	D.2.9 Management	
software reliability	D.2.4 Software/Program Verification	infrastructure consultancy
Software Quality Assurance	D.2.9 Management	Development
Software quality	D.2.4 Software/Program Verification	System development (design - programming - test)
smidige metoder i store prosjekter	D.2.9 Management	
Simplicity	N / A	data structure design; data conversion
simplicity	N / A	Time estimates, design and development

Security	N / A	programming
Scalability	D.2.7 Distribution, Maintenance, and Enhancement	Zimbra
Safe multi threaded programming models	D.2.3 Coding Tools and Techniques	ESI
reliability	D.2.4 Software / Program Verification	system deployment
Reducing complexity of software development	N / A	Development
Quality control	D.2.4 Software / Program Verification	TDD
product direction vs. development direction	D.2.9 Management	design software architecture
Practical problems	N / A	Programming
Practical implications	N / A	System maintenance
Phasing out large legacy systems	D.2.7 Distribution, Maintenance, and Enhancement	A&D
Persistent Object Storage	D.2.3 Coding Tools and Techniques	System administration
Performance	N / A	integration
OpenSource as movement, as opposed to being bought by large corporations as IBM and Oracle	N / A	At the time of writing, my main focus is maintenance of legacy code.
Multithreading performance	N / A	Web developing
multithreading	N / A	Java JEE5 development (EJB3)
Multicore	N / A	Programming
Mashups, cload, new languages on JVM	N / A	Scala, web/integration, REST
JBoss clustering in relevance / as an example of to advanced topic of distributed computing (e.g. group communication)	N / A	Maintenance, self-stud
integration	D.2.12 Interoperability	
increased automation	N / A	QA
Importance of methodology	D.2.9 Management	Development (Java/J2EE)
Impact of iteration length and delivery frequency on project success	D.2.9 Management	Coaching teams delivering internal GUI application in technology and practices

How to make software projects deliver higher quality	D.2.9 Management	
how to make businesses understand the vital importance of having technicall savy project managers in development projects	D.2.1 Requirements / Specifications	Developing applications in Java/GWT
How the real world works	N / A	Programming
Get industry practice	N / A	Software Maintenance, Programming
Free and open source software	N / A	Using the software for myself and my customers.
Formal methods	D.2.4 Software/Program Verification	Web developing
F/LOSS	N / A	Web development
Efficiency & ROI	N / A	Sales & Marketing
Effective practices for developing well tested products	D.2.5 Testing and Debugging	Delivering / developing well tested products....
dynamic languages in large scale projects	D.2.3 Coding Tools and Techniques	Programming / architecture
dynamic languages (practise)	D.2.3 Coding Tools and Techniques	development / architecture (integration)
distributed software dev.	D.2.9 Management	devel.
development processes	D.2.9 Management	Project leader
Developer-customer communication	D.2.1 Requirements / Specifications	Sales
Context awareness	N / A	
concurrency and parallellism	N / A	integration
Concequences of mixed technology use (all apps have many diffrent technologies in them chosen at "random" based on previous experiance or what is knowin in the deciction moment)	D.2.12 Interoperability	Development - maintanence, porting old to new technology.

Compiler time	N / A	
clustering	N / A	linux system administration
Better documentation	D.2.7 Distribution, Maintenance, and Enhancement	course material
Attitudes and tradition. Like this! We make a lot of bad choises and need to be told!	N / A	Making accelerometer controls usin Sun SPOS and Wiimotes. JavaFX
Application of software engineering practices in free software	D.2.9 Management	management of development
Agile development methods	D.2.9 Management	Integration
Agile development	D.2.9 Management	Java development
agile and lean methods	D.2.9 Management	programming, maintenance
A replacement for SOA (or improvement...)	D.2.11 Software Architectures	Billing Applications for the telecom industry
(Double) blind tests	N / A	programming
web application frameworks (get rid of ajax)	D.2.11 Software Architectures	Java JEE5 development (EJB3)
the engineering aspects of standardization	D.2.3 Coding Tools and Techniques	integration
Testing of multi threaded programs, performance and concurrency	N / A	ESI
Testing	D.2.5 Testing and Debugging	
team productivity	D.2.9 Management	design software architecture
support / drift / deploy	D.2.7 Distribution, Maintenance, and Enhancement	project planning resource
Standards of integration	D.2.12 Interoperability	Billing Applications for the telecom industry
Standards	N / A	Web development
Space/time metaphors in programming / computer usage	N / A	System administration
Software quality vs requirements	D.2.5 Testing and Debugging	Delivering / developing well tested products....

Semantics on the web	N / A	Developing web applications
Scientific measuremens of the benefits of Agile Development (if there they are there)	D.2.9 Management	Development
reducing complexity	D.2.11 Software Architectures	infrastructure consultancy
Quantifying advantages/disadvantages	N / A	programming
Quality and testing	D.2.5 Testing and Debugging	Testing
project management	D.2.9 Management	devel.
Programming language comparisons	D.2.3 Coding Tools and Techniques	Development (Java/J2EE)
Productivity	D.2.9 Management	TDD
Prioritazation of tasks vs. actual tasks chosen to do. Non importen tasks often get more time tacitly.	D.2.9 Management	Development - maintanence, porting old to new technology.
performance	N / A	linux system administration
Ordering of words	N / A	System maintenance
Open data sources on the web	N / A	Scala, web/integration, REST
Making virtal machines more effecient	N / A	
Make interesting research more available - show up to conferences and speak about it in a non-scientific manner, make code available in a Maven repository, and so on	N / A	Programming
Load Balancing	N / A	Zimbra
Learning	N / A	Sales & Marketing
Integration	D.2.12 Interoperability	project planning resource
How to write good specification	D.2.1 Requirements / Specifications	Development
How to make software systems adaptable, and flexible for future changes.	D.2.7 Distribution, Maintenance, and Enhancement	Java development

Globally distributed services	D.2.11 Software Architectures	Using the software for myself and my customers.
Formal methods	D.2.4 Software/Program Verification	Temporarily laid off (permittert)
facial recognition	N / A	programming
Estimation	D.2.9 Management	Sales
Empirical measurements of (working with) scripting languages: Perl vs Python vs Ruby etc.	D.2.3 Coding Tools and Techniques	Maintenance, self-study
Efficiency	N / A	Time estimates, design and development
durability	N / A	data structure design; data conversion
development tools	D.2.6 Programming Environments	Project leader
Development teams	D.2.9 Management	System development (design - programming - test)
Developers and project owners need good metrics to assess complexity of technology stacks and codebases to improve quality of software engineering processes	D.2.8 Metrics	Developing applications in Java/GWT
Benefits and drawbacks from off-shoring	D.2.9 Management	Integration
Balance between team level and enterprise level decisions	D.2.9 Management	Coaching teams delivering internal GUI application in technology and practices
automatization	N / A	development / architecture (integration)
automation of processes (build, testing, deployment)	D.2.7 Distribution, Maintenance, and Enhancement	programming, maintenance
Algorithms and ways of improving code itself, as opposed to focus om processes	D.2.3 Coding Tools and Techniques	At the time of writing, my main focus is maintenance of legacy code.

Agile methods	D.2.9 Management	Making accelerometer controls usin Sun SPOS and Wiimotes. JavaFX
Video over Internet	N / A	Developing web applications
User interfaces	N / A	Making accelerometer controls usin Sun SPOS and Wiimotes. JavaFX
standardization	N / A	programming
Self-programming programs	N / A	Using the software for myself and my customers.
Scability	N / A	linux system administration
Realtime Systems	N / A	Billing Applications for the telecom industry
Realtime shared computer interaction	N / A	System administration
Project management	D.2.9 Management	Sales
Programming	D.2.3 Coding Tools and Techniques	System maintenance
Productivity (short and long term) of dynamic vs. static languages	D.2.3 Coding Tools and Techniques	Integration
Patterns / Best practises	D.2.11 Software Architectures	Sales & Marketing
Open Source software compared to proprietary software	N / A	Development (Java/J2EE)
More (surveys) on (practical) modeldriven development techniques	D.2.2 Design Tools and Techniques	Maintenance, self-study
Intuitivity	N / A	Time estimates, design and development
interoperability	D.2.12 Interoperability	infrastructure consultancy
I would like to see research proving that software developers need quiet and comfortable working areas to be productive. How can one solve hard problems while sharing a floor with 10-15 other slave-developers buzzing with daily scrums and what not.	N / A	Developing applications in Java/GWT

How to write good test scenarios	D.2.5 Testing and Debugging	Development
how to design for availability and performance	D.2.10 Design	programming, maintenance
High Availebility	N / A	Zimbra
functional programming	D.2.3 Coding tools and Techniques	Java JEE5 development (EJB3)
Everchanging updgrades of all technologies. Cost of never knowing someting well before it changes into someting new.	N / A	Development - maintanence, porting old to new technology.
Distributed computing	N / A	At the time of writing, my main focus is maintenance of legacy code.
Development environment	D.2.6 Programming Environments	System development (design - programming - test)
Cost	D.2.9 Management	Testing
concurrency	N / A	development / architecture (integration)
Business - developer communication	D.2.1 Requirement / Specifications	Java development
Better operating systems	N / A	
Best practises	N / A	Web development
Auto-documentation	D.2.7 Distribution, Maintenance, and Enhancement	TDD