ANCHORING AND ADJUSTMENT IN SOFTWARE ESTIMATION

by

Jorge Aranda

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

ANCHORING AND ADJUSTMENT IN SOFTWARE ESTIMATION

Jorge Aranda

Master of Science

Graduate Department of Computer Science

University of Toronto

2005

Software estimation research is normally concerned with designing models and techniques that help estimators reach accurate effort calculations. However, since software estimation involves human judgment, we should also consider cognitive and social factors that have been observed in psychological studies on judgment. A potentially relevant factor is the anchoring and adjustment cognitive bias. It takes place if, when attempting to respond a complex question, the respondent is given a possible –though quite likely incorrect- answer. The respondent adjusts it internally to reach a more plausible answer, but the adjustment is frequently insufficient.

This thesis presents the results of an experiment about anchoring and adjustment in software estimation. Results show that anchoring and adjustment changes the outcome of software estimation processes. They also suggest that estimators tend to have too much confidence in their own estimations.

Acknowledgements

This thesis would not have been possible without the extraordinary support, advice and encouragement of Steve Easterbrook. As my supervisor and teacher he made me feel it is fine to reach out from Computer Science to other fields in order to enrich our own. His confidence and sensible advice inspired me to develop my own ideas and see them come to fruition, which has been enormously satisfactory.

I would like to thank Eric Yu as well, both for the invaluable comments and leadership during our research, and for accepting the responsibility of being the second reader of this thesis. It has been an honour for me to work with him, and I hope I will have the opportunity to keep learning from his insights and character.

Greg Wilson provided me with many helpful comments, stimulating ideas and much needed contacts for experiment participants. I am indebted to him for all this. I also thank Mehrdad Sabetzadeh for his valuable time and advice on how to write a thesis.

Special thanks should go to all participants of this study. Their contributions were vital – I could not have done anything if they had not spent some time working selflessly in this problem.

I met Marcel Leica and Jonathan Amir on my first day at the University of Toronto, and I have been fortunate to have their friendship ever since. I thank them, the great people I have met here, and my friends in Mexico for making it easier to be away from my home country.

My father has always been within reach, advising me and giving me his arm to hold whenever things get rough. My mother, although no longer here, keeps returning to my memory with loving examples on how to live well.

Finally, my deepest gratitude goes to my wife and accomplice, Valeria. Her encouragement, enthusiasm and love are the greatest gifts I could wish for. She gives meaning to it all, and these words are not nearly enough to give her the credit she deserves. Thank you, Vale. This thesis is for you.

Contents

Table of Figures

Chapter 1

Introduction

Software estimation is a problem that has attracted a considerable amount of research within software engineering, but it has so far failed to provide either a consensus on the best approach to estimation or a clear process to produce consistently accurate effort estimates [Kem87], [BAC00]. A possible reason for this lack of powerful, satisfactory techniques is that software estimation is frequently approached with the assumptions that it is, in essence, a technical or mathematical problem [Dol01], that it can be automated, or at least standardized [DeM82], that vital, unknown pieces of information can be accurately approximated at the early stages of a project's lifecycle [Boe81] and that the estimation process can be kept separate from organizational politics, marketing or business cycles and external influences.

Estimation can, and should, be approached from another angle as well: from the human judgment branch of psychology. Software estimation is, after all, performed by humans, and it is concerned with human activities; it is done under uncertainty and within a social setting that alters the behaviour and performance of those involved. Several studies [Jør04] strengthen the validity of exploring software estimation as an inherently human activity, subject to cognitive and social effects.

Adopting the psychological approach, this thesis is interested in the effects of judgmental bias, and specifically of anchoring and adjustment bias, in software estimation tasks. Anchoring and adjustment is a widely observed and documented phenomenon [MS01] that occurs when people face choices under uncertainty and the result of the choice can be expressed as a number within a range. If judgment of the matter is difficult, we tend to grasp an anchor, which is a tentative, even if unlikely, answer; and we adjust it up or down according to our intuition and experience to reach the final result. The adjustment, however, is frequently insufficient to compensate for the biasing effects of the anchor, and the final answer is distorted due to this influence.

If the effects of anchoring and adjustment are observed in software estimation processes, then this heuristic deserves a deeper consideration than it has had in the software estimation field. This possibility motivated an experiment that consisted of a software estimation exercise designed to detect anchoring and adjustment effects in estimators, which is documented in this thesis.

The thesis is structured as follows. Chapter 2 surveys the field of software estimation research, and explores the nature of software estimation as a human judgment activity. Chapter 3 provides a similar survey of the field of judgmental bias, and specifically of anchoring and adjustment and of research relating human judgment and software estimation. These two chapters together provide the fundamentals and related work necessary to frame the findings of the thesis within research in both fields.

The research questions that motivated the study are detailed in Chapter 4. Chapter 5 describes the design of the experiment, and Chapter 6 describes its execution. Results and analysis can be found in Chapter 7. Finally, Chapter 8 closes with a discussion based on the findings of the experiment and concludes the thesis.

Chapter 2

Software Estimation Fundamentals and Related Work

Time, cost and effort estimation for software projects has been a thorn in the side of software engineering since its beginnings [Bro95], [Sta94]. On one hand, software projects frequently need concrete estimation numbers on their early stages in order to take proper managerial decisions; on the other hand, reliably obtaining those numbers at that time is still risky and technically unfeasible. Boehm et al. [BCH+95] report that estimating a project on its very first stages yields estimates that may be off by as much as a factor of 4. Even at the point when detailed specifications are produced, professional estimates are expected to be wrong by +/-50%.

Significant research has been devoted within software engineering to design estimation techniques that increase estimates reliability. Numerous publications have addressed this issue, proposing mathematical models for estimation, attempting to understand the mental processes that are involved in the minds of experts, and questioning whether it is even possible to obtain accurate estimates early in the software development lifecycle.

This chapter explores fundamental work on software estimation. It does not intend to be a complete survey of the field, but to present some basic findings and categorizations, and to set the ground for the following discussion on biases in human judgment.

2.1 – Nature of Estimation

Before reviewing any software estimation techniques, we should set a definition for “estimates”. Interestingly, most published papers assume an agreement on the meaning of estimates, but there are reasons to believe this assumption is unjustified. Grimstad et al. [GJM04], for example, call for further clarity in estimation discussions, and give several possible meanings to the concept.

On its most basic sense, an estimate is a prediction of how much effort, time or cost is necessary to complete a task. Software estimates, particularly, are complicated because the variability involved in software development prevents from giving accurate and precise predictions.

It is best to think of estimates as possibilities. Consider the distribution curve in Figure 2.1. For every development task there must be an absolutely minimum possible time it takes to be completed [Arm02], it is impossible to reduce it further. It is also safe to assume that, unless the task is impossible, a reasonable upper bound can be given for its completion. From the lower to the upper bound, all points are possible durations for the task.

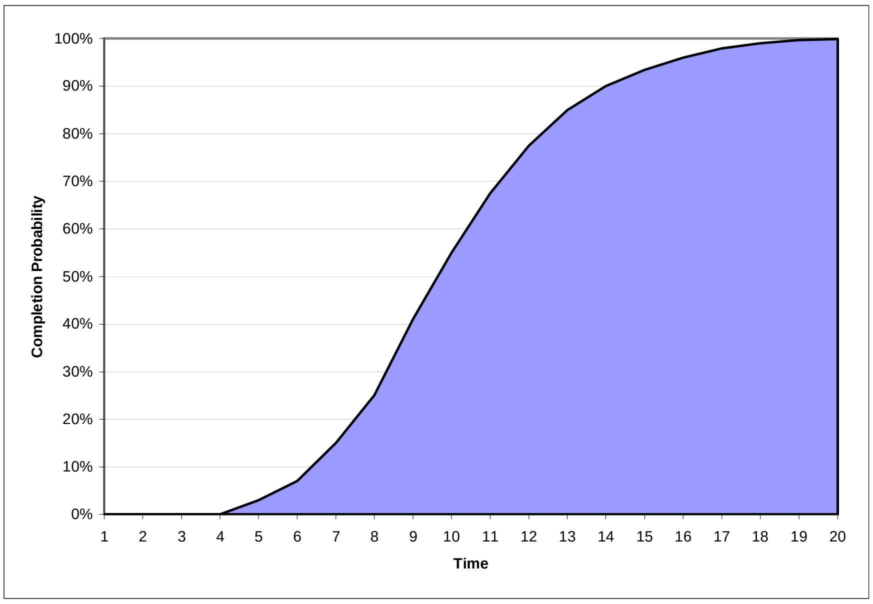

Figure 2.1 – Completion Probability Distribution

Considering this curve, DeMarco [DeM82] believes that the default definition of estimate among professionals is “the most optimistic prediction that has a non-zero probability of coming true”, that is, a quantity that is just above the lower bound of the range of possibilities previously discussed. Most developers aim for this figure when estimating, ignoring risks and external influences. DeMarco argues that a correct definition of an estimate should be “a prediction that is equally likely to be above or below the actual result”, that is, the point at which the probability of having completed the task is 50%. This seems to be the concept most researchers have in mind when they talk of estimation. If this was the meaning we assign to estimates, about half of software projects should be completed before the estimate, and half would be overtime. This definition, however, is not likely what managers have in mind when they ask for an estimate, since the business risk of a 50% probability of missing a deadline may be too high. Managers would look for a more reliable figure, something around the 80% probability, or higher.

Estimates are not always thought of as probability predictions. Frequently, estimates are cognitively equivalent to development *budgets* [GJM04]. This is unfortunate: if financial plans are made based on estimates and, as discussed above, 50% of software projects are completed after their estimates, then all of them will be over budget, and some will be probably cancelled due to lack of funds.

Yet one more meaning of estimate turns the definition around. There is an allotted amount of effort (or time, or cost) allowed, so what is estimated is the development *plan* [Jør04b].

Up to this point we’ve assumed the result of an effort estimation process is a number, in work hours, months or cost, and not an interval. However, prediction intervals are a better reminder of the uncertainty involved in estimation [Jør04]. Interval estimates transmit the idea that the prediction may still be too vague on the early stages of a project, and intervals narrow down as certainty increases and the project progresses [BCH+95]. Estimate intervals can be expressed as a [minimum, maximum] pair or as a base estimate with a +/- percentage margin of error.

Summarizing, estimates can be thought of as predictions of minimum effort, predictions of average effort, risk control mechanisms, budgets and development plans. Estimates can be single numerical points or intervals. In this thesis we will use the meaning of estimates as *predictions of average effort*, that is, predictions that are technically just as likely to be above or below the real outcome; and we will explore both the single numerical point and the interval representation of estimates.

If we consider estimates as predictions, we should explore the prediction process that estimators follow. How do humans reach predictions, how do they estimate? We should turn to the behavioural sciences for the answer, but the matter has been studied there without much success: "Psychological research on real-world quantitative expert estimation has not culminated in any theory of estimation, not even in a coherent framework for thinking about the process" [BS93].

It is not surprising, considering the difficulty of producing a theory of estimation based on human judgment, that computer scientists prefer to create mathematical estimation models. But as we will discuss in the next section, the efficacy of models, at least size-based models, is doubtful. The LOC–effort relationship does not hold well enough [Dol01] and estimators are better at estimating effort than size –which cancels the benefit of size-based estimations [HH91], although they generally do not seem to be very good at either.

Moreover, estimators do not think of program size naturally. In a survey of estimation practice, Hihn and Habib-agahi concluded that all estimators attempt to predict effort, but only 49% of them estimate size [HH91]. Furthermore, only 22% use size estimates as part of their estimation process (the remaining 27% estimate size because they are required to by their companies, and do not incorporate that estimation into their results).

The same survey reports that model-based estimation is very rarely used by professional estimators. Only 7% of respondents use models as their primary estimation method. An additional 11% use models secondarily. Apparently, estimating experience appears to be a factor on the preference of estimation techniques. The less experienced software estimators have a greater likelihood of using models as their primary method. More experienced software estimators tend to switch to analogy and expert-based techniques.

Finally, any discussion of estimates should consider that they are part of a soft and complex system in which subtle factors may alter the outcomes considerably. One notorious factor is the variation in productivity of developers. According to some studies [DL99] the best programmers are 10 times more productive than the worst programmers, and 2.5 times better than the median. Programming teams also have widely diverse performances. Estimation techniques and models attempt to account for productivity variations, but they commonly fail to reflect upon the impact of this factor on the general results of a project.

Another elusive factor lies upon the field of requirements engineering. Badly stated, missing and changing requirements can increase a project's effort to several times its expected value. Software requirement issues account for half of the top ten risks for software projects, with the potential of extending a project to several times their intended time and budget. Estimation techniques rarely include such considerations explicitly in their processes.

Even seemingly unrelated events and simple observations produce changes in performance. According to Abdel-Hamid and Madnick [AM86], estimates themselves are a factor in the real effort of software projects. That is, estimates may become self-fulfilling prophecies as developers struggle to meet the results that are expected from them. If this is true, and there is no indication on the contrary, a low or a high estimate (which, we should remember, intends to be only a prediction) may influence the project's development in unexpected ways.

We should remember that software development is an activity that needs a high degree of creativity, inventiveness and social interaction. It is extremely difficult to simplify such abstract and subtle elements to a satisfactorily precise estimation model. The next section describes the most relevant attempts to address this problem.

2.2 – Estimation Techniques

In this section we will survey most currently accepted software project estimation techniques. This survey is not exhaustive, and is provided to help locate the types of estimation performed in the experiment this thesis addresses within an estimation techniques framework.

We will base this survey in Boehm, Abts and Chulani estimation techniques classification [BAC00].

2.2.1 – Model-based Techniques

Judging by the amount of research devoted to them, model-based techniques are probably the most popular approach among academics. The number of papers proposing, refining and reinventing model-based estimation techniques is overwhelming, although the principles behind them are consistent and relatively simple.

The core of model-based techniques lies in the assumption that a reliable mathematical model to calculate the effort necessary to develop software may exist. Research in model-based techniques focuses on attempts to discover such a model [Dol01].

The size of the software being developed is generally considered to be the primary factor for the effort it takes to develop it: a small application naturally demands less effort than a large one. Size is normally calculated a a number of lines of code or functions needed in the system. However, size is not the only factor for effort, so other drivers need to be accounted for. A common basis for a model for software estimation is the equation:

$$\mathit{Effort} = \mathrm{A}\ \mathit{Size}^{\mathrm{B}}$$

In the equation, A and B are constants obtained after considering factors such as development experience, reliability requirements and domain knowledge. The exponent B is generally higher than 1, so the equation indicates exponential growth, although B does not normally stray too far from 1.

It is worth noting that this equation is not universally accepted [Dol01], and that even within techniques that use the equation, the means to define A and B vary. The meaning of *Size* is sometimes discussed as well.

Some of the most widely known model-based estimation techniques are explained below.

- **SLIM – Software Life-cycle Model:** Developed by Putnam and Myers [PM92]. It is based on the application of the Rayleigh distribution curve to determine the effort needed in a software project, which varies during its lifecycle. Although SLIM was one of the first accepted estimation models, it is not commonly used these days.
- **ESTIMACS:** This proprietary model-based technique, exposed by Rubin [Rub83], follows a business model approach. The input to the model is a set of answers to 25 questions referring to several cost drivers. Its proprietary status prevents a greater spread in its use.
- **COCOMO:** The Constructive Cost Model was developed by Barry Boehm [Boe81] and it is arguably still the most popular and referenced estimation model published. It is based on the equation stated above, as most pure model-based techniques, and it has three complexity levels. They differ in the detail given to the calculation of factors that modify the exponent of the equation. The *Size* of the application to be developed is expressed in lines of code (LOC). Boehm warned that his model may need to be calibrated to reflect the details of the estimator organization, and according to third-party validations [Kem87], calibration is vital for even passable estimates with COCOMO. The model was created when the waterfall lifecycle for software development was considered the standard methodology for software projects, and is therefore outdated in its original form. There has been research in trying to evolve the model along with current software development approaches. For example,

Benediktsson et al. [BDR+03] present a COCOMO-based model for iterative and incremental developments, and Boehm updated his model to obtain COCOMO II.

One of the common arguments against COCOMO –and all LOC-based models– is that lines of code are not a proper measure of the magnitude of the effort needed to develop software [Arm02]. According to Armour, "using (lines of code) as a measure of knowledge quantity is pretty much like weighing a book to figure out how much knowledge it contains". Capers Jones [Jon96] argues that not only lines of code are not the best metric –they are downright misleading: they tend to hide productivity gains, inhibit code reuse and cause bad development practices. However, the best argument against the use of lines of code as a basis for software effort estimation is that professionals estimate lines of code with less accuracy than estimates of effort [HH91].

A different metric –function points, FPs – was developed at IBM [Alb79] with the goal of addressing these issues. Function points represent the functionality of a program, and they seem to be more intuitive than lines of code [Jon96].

Two examples of model-based techniques with FPs are:

- **Checkpoint:** Checkpoint is a proprietary tool from Software Productivity Research (SPR) developed by Capers Jones [Jon96]. It is based on a calculation of FPs (inputs, outputs, displays, queries, files) for the software to be developed, which is modified with a factor that considers experience, productivity and several other project characteristics.
- **COCOMO II:** Although COCOMO II [BCH+95] is not exclusively FP-based, one of its most notorious changes from the original COCOMO is the possibility of estimating effort based on the functionality of the program to be developed, instead of on lines of code. Most of the characteristics of the original COCOMO are still featured in this later incarnation, such as the three estimation complexity classifications and the set of project drivers that modify the pure FPs estimate.

Most of the models discussed so far were developed using statistical regressions from project data, but have rarely been properly validated in real use. In one of the first studies attempting to validate the performance of model-based techniques, Kemerer [Kem87] reviewed the results of estimates produced with SLIM, COCOMO, a generic function points model and ESTIMACS. He gave his own study an advantage that practitioners do not have: he knew beforehand the size, in lines of code, of the software he was estimating, and he used it as an input to his calculations. Practitioners must estimate this input to reach a result. Even with this advantage, he found that, without calibration, models have a disproportionately bad performance. Depending on the model used, average results were from 103% to 772% off from reality. Another, more recent validation study [JRW00] produced similar conclusions.

There is another aspect of working with model-based techniques: the question of whether estimates from the model should be modified by estimators according to their experience or if they should be left untouched. Subramanian and Breslawski [SB95] conducted a study that concluded that estimates that are generated from models and subsequently modified according to experience are more accurate than if they were left unmodified.

Dolado addresses the issue of the existence and possible nature of a software effort function [Dol01]. According to Dolado, academics have proposed linear, quadratic and exponential functions to explain their software effort data without having a theoretical background justifying any of those alternatives. In his study, data reported for twelve software estimation studies are merged to attempt to identify if those data correspond to a clear mathematical equation. Results show that, although there apparently is a relation between size and effort, the spread is so marked that it is not advisable to treat program size as a defining factor of development effort. In his words: “Regardless of the method, the basic size-effort relationship does not show satisfactory results.” He concludes: “The present state of the art in software estimation does not provide a theory that explains all the data available”.

2.2.2 – Learning-oriented Techniques

It is dubious whether learning-oriented techniques form a category by themselves, since it is hard to separate them from expert-based techniques (which are explored in the next section): Experience and learning are intertwined. However, at least one learning-oriented technique is concerned with representing learning through artificial neural networks, and another technique has a standard approach to the incorporation of past experiences in estimating decisions; therefore, the learning-oriented category is considered worthy of attention as a separate category. Learning-oriented techniques are based on the assumption that past performance is a good indicator of future results. They generally produce satisfying results when the project being estimated resembles previous projects, but bad results when it deals with new applications, domains or practices.

The following two are the most common learning-oriented techniques for software effort estimation:

- **Analogy-based estimations:** Shepperd and Schofield [SS97] propose using data from previous projects to estimate how much effort will the next demand. This approach is especially recommended if the development team has dealt with projects with similar scope and/or domain. The technique has room for computerized intervention: If there is enough data from in-house projects, the use of computerized tools would detect which previous projects are better suited as analogies to any new project with defined characteristics. Using such tools would help reduce human bias when picking sources of analogy. But in an empirical study, Walkerden and Jeffery [WJ99] compared analogy-based estimations with and without the aid of computerized tools. Results indicate that people choose sources of analogy better than tools; which suggests that either human bias is better than automated processes, or that present day tools need refinement.

- **Neural networks:** There have been at least two attempts to approach the problem of software estimation using neural networks ([GM96] and [FWD97]). Neural networks are attractive in software estimation because the impact of each relevant factor in a software project is not known, and it might be best to allow a neural net to adapt to data as it becomes available in order to improve estimates quality. However, Boehm et al. [BAC00] point out that neural networks in software estimation are still an immature approach, and more time will be necessary in order to assess their true efficacy.

2.2.3 – Expert-based Techniques

Expert-based techniques are, at the same time, the most widely used estimation methods [HH91] and the ones with arguably the worst standing among academics [Hug96]. Most research in software estimation deals with proper models to perform estimations, not with an analysis of the way experts reach their conclusions. Three notable expert-based techniques are:

- **Delphi:** Named after the Greek Oracle, this technique depends on the work of a group of experts that attempt to reach a converging estimate. In its basic form [Hel66], the Delphi technique has a group of experts working separately to produce an effort estimate. Their individual results are made known to the others, and they are subsequently allowed to review their own estimate. If no agreement is reached after the second estimate, the average of their individual estimates is taken as the final result of the process. A modified Delphi technique, called Wideband Delphi [Boe81] allows the individual experts to communicate among themselves to share the reasoning behind their results, ideally being able to better adapt them. Supporting the Delphi technique, there is evidence that group estimation decisions are better than individual decisions. For example, Moløkken-Østvold and Jørgensen [MJ04] report that groups of estimators are generally less optimistic than individuals, and therefore we should expect longer –perhaps more realistic- estimates from them.

- **Work breakdown structure:** Also called bottom-up estimation, a work breakdown structure (WBS) estimates the effort needed to develop a project by adding the time it takes to develop each of its components [Bai89]. It is supported by the ideas that it is easier to estimate the effort necessary to perform a simple task than a complex task, and that errors in small estimates are balanced and cancel each other when considering concentrated estimates. WBS is generally considered a variation of the next estimation technique.
- **Freeform expert estimation:** It is clear that expert estimation is used in the great majority of software estimation processes. As was mentioned before, Hihn and Habib-Agahi [HH91] report that 83% of the Jet Propulsion Laboratory estimators use informal analogies as their primary estimation technique. In contrast, models are used as the primary technique by only 7% of estimators. Other studies confirm this tendency. Heemstra and Kusters [HK91] found "intuition and experience" as the basis for 62% of estimates of the projects they studied, and 16% of estimates were based on formalized models. The percentage of expert estimation use in [Pay96] is 86%, 72% in [KPM+02], and 84% in [Jør97]. There are reports of large companies where the percentage of model-based estimations is zero [HTA00]. Expert estimation has been found to be the most commonly used estimation technique in both [HH91] and [Hug96]; although some researchers do not call it a technique, but merely "guessing". Indeed, its basic feature is that there is no defined process or approach to perform an estimate. Experts are assigned the responsibility of reaching an estimate by whichever means they see reasonable. The criteria to call estimators "experts" is quite loose. An expert estimator could be someone with only academic knowledge of software engineering, or it could be a person with years of experience and knowledge to draw from. Incidentally, the amount of experience that estimators have is not a good indicator of their accuracy [JS04]. As was mentioned before, expert estimation is commonly shunned by

academia, possibly because of its lack of sophistication. In an extensive survey, Jørgensen [Jør04] points that there are few published studies about expert estimation in the field of software development –although it is a topic common in the behavioural sciences. The scarcity of papers about expert estimation should not be considered an indication of their effectiveness, or lack thereof. In the same survey Jørgensen points that in the 15 studies found comparing expert estimation with model-based approaches, models outperformed experts in 5 occasions, experts were more accurate in 5 more, and there was no clear preferred technique in the remaining 5. Therefore, the possibility exists that expert estimation is being underestimated as a valid technique.

2.3 – Estimation as a Human Activity

As was discussed previously, expert estimation has an unfavorable standing among researchers, but wide acceptance among practitioners. In this section we will delve deeper into this subject, exploring whether expert judgment –or, more appropriately, human judgment- can truly be separated from estimation processes.

There are several opinions as to what does it mean to perform "expert estimation", and on what it is to be an expert. Jørgensen [Jør04] uses a rather broad definition for expert estimation: it is the result of applying estimation strategies in the interval from unaided intuition ("gut feeling") to structured estimation (supporting expert judgment with historical data, process guidelines and checklists). The defining characteristic of expert estimation is, according to Jørgensen, that "a significant part of the estimation process is based on a non-explicit and non-recoverable reasoning process, i.e., 'intuition'."

Other researchers favor a narrower meaning of expert estimation, attempting to restrict who can be an expert or defining a basic set of semiformal processes experts need to follow, such as the Delphi technique [Hel66].

However, it is worth questioning, as Hughes does [Hug96], if there actually exists, or can exist, an estimation technique that does not rely on human judgment. Most accepted estimation techniques, with the exception of model-based techniques, depend heavily on the judgment of estimators. Analogy-based estimation relies on experts choosing proper sources of analogy. Hybrid approaches and work breakdown structure analyses are inherently judgment based.

Model-based techniques attempt to hide their reliance on expert judgment, but not to eliminate it. At the heart of a COCOMO estimation, for example, lie two judgmental decisions that are needed every time an estimate is produced: an approximation of the number of lines of code that the software will have, and the subjective weights assigned to several cost drivers that will modify the outcome of the estimation equation. A human needs to take those decisions, therefore, human judgment is deeply involved in model-based techniques as well.

According to Pengelly [Pen95], the efficacy of formal software estimate models is dependent on the good judgment of the estimators, since they require expert estimates of important input parameters. This reliance on expert estimation is not exclusive of software development; it has been observed and naturally accepted in a variety of settings [BH90].

Software estimation’s reliance on human judgment is frequently understated. Mathematical models for effort estimation convey the idea that forecasting is a clean, defined process with little interference from psychological and social factors. But human judgment in effort estimation exists, and plays a vital role in every particular estimate. Understanding the implications of this reliance on judgment should be a primary objective of software estimation researchers.

Chapter 3

Anchoring and Adjustment Fundamentals and Related Work

Anchoring and adjustment is a cognitive bias, and to explain it we should connect to research in psychology, and specifically to the area of judgmental bias. In this chapter we will explore what are judgmental biases, and then we will focus on anchoring and adjustment concretely. After a brief survey of both topics we will cover work that relates the areas of human judgment and software estimation.

3.1 – Judgmental Bias

Judgmental bias is considered to be any deviation from reality that prevents the objective consideration of a situation [Hog80]. Although this simple definition carries some important philosophical assumptions –mainly that there is one, unique reality and that it is possible to consider it objectively- it is useful to think of bias using this definition in the following discussion.

Several types and sources of bias in human thought have been identified, and in order to classify them it is helpful to consider a conceptual model of judgment provided by Hogarth [Hog80], which is depicted in Figure 3.1.

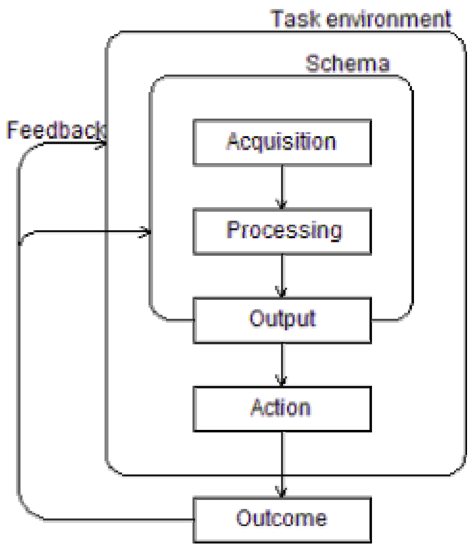

Figure 3.1 - Hogarth's conceptual model of judgment

Hogarth situates the person under analysis, and the person's schema, inside a task environment. In this model, a person acquires information from the environment, processes it, and outputs a resulting action. This action produces some sort of feedback that affects the environment and the person's future actions.

Using this model helps to discuss the types of bias humans are subject to. We can locate the possibility of bias in the Acquisition, Processing, Output and Feedback conceptual stages. The following subsections list and explain the types of bias that may occur in each of them. Note that the lists and most of the discussion in this section are adapted from Hogarth [Hog80] and Kahneman, Slovic and Tversky [KST82]. While studying these lists it may be helpful to maintain software engineering, and specifically software estimation, in mind to see if and how these biases are likely to present themselves in such domains.

3.1.1 – Information acquisition biases

The problems of bias in information acquisition generally refer to the *saliency* of the information that is acquired. Recent, representative and/or believable data, for example, is recalled and acquired with greater ease than the rest. It is important to note that memory is not an accurate recording of previous events: different persons may have completely different memories from the same event depending on the things each was biased to perceive. Some of the common biases in information acquisition are:

- **Availability:** Tversky and Kahneman [TK82] have shown that humans tend to estimate the likelihood of an event and the frequency of occurrences of a class by assessing the ease with which the relevant mental operation of retrieval, construction or association can be carried out. Classic examples of this heuristic are that people estimate that the letter *R* appears more frequently in the first than in the third position of English words, which is incorrect; and that after studying a list of names where the persons of one gender are more famous than those of the other gender (which in turn are more numerous in the list), people think that the former list is longer than the latter.
- **Risk perception:** This is a specific subclass of availability bias. Slovic et al. [SFL82] proved that people tend to assign a greater probability of risk to more publicized and recent dangers while greater but silent risks are underestimated.
- **Selective perception:** People tend to perceive information they expect to perceive, and downplay or disregard conflicting evidence.
- **Concrete information:** We tend to remember information that was concretely given to us (for example, face to face) better than abstract information (like statistical base rates). For example, when considering to buy a certain car model we will likely give more thought to

the direct advice of a friend than to each of the 100 respondents to a survey in a specialized magazine.

- **Data presentation:** The way people receive information biases its acquisition. The first and last items in a sequential presentation tend to be given greater relevance than the rest. Also, apparently thorough and complete information blinds people to critical omissions in its exposition.

3.1.2 – Information processing biases

Information processing biases are caused by the decision making mechanism people use. It is probably the judgmental stage in which most types of bias occur. Most of the problems in processing information arise from the complexity of the data to process, the unwillingness to spend mental effort and the lack of consistency in judgment. The following list expands these concepts:

- **Anchoring and adjustment:** Please see the description in section 3.2 for a discussion on the topic of anchoring and adjustment.
- **Inconsistency:** Most people are unable to apply a consistent judgmental criterion over a repetitive set of cases.
- **Conservatism:** People have been shown to fail to adapt or revise their opinions in the face of new information.
- **Non-linear extrapolation:** We have an inability to extrapolate exponential growth processes or to calculate the conjunctive probability of several events, which we tend to overestimate.
- **Habits and rules of thumb:** This may be classified as a subclass of availability biases. A previously tried “good enough” choice may cause the automatic elimination of every other –possibly better- alternative in subsequent events.

- **Representativeness:** Kahneman et al. [KST82] say that humans tend to mix representativeness with probability: When classifying a piece of information, we are likely to assign it in a class on which we believe it typically belongs, not in the class on which it statistically belongs more often. For example, people predict that an intelligent, shy, science fiction fan student is enrolled in computer science, even when they have previously shown that they know that, according to student enrollment distribution, it is more likely for *any* student to be enrolled in a humanities or education program, regardless of personality traits.
- **Worthless data:** Having no specific data on a subject is better than having worthless data [KT82]. When there is no specific data, people rely in base rate information. When there is worthless data, people ignore base rates and try to give meaning to what they have.
- **Law of small numbers:** This is probably one of the best known heuristics discovered by Tversky and Kahneman [TK82b]. Characteristics of a small sample are expected to be representative of the population from which they were drawn. (So six consecutive coin tosses resulting in “heads” are classified as weird and non-random).
- **Justifiability:** According to Hogarth, when provided with an apparently rational argument people may simply accept and get along with it, even if it is rationally inappropriate.
- **Regression:** Statistical regression is not a bias, but failing to attribute exceptional cases to it is. Kahneman and Tversky [KT82] narrate a case of inability to understand regression. In an army, flight instructors held the belief that the performance of their students improved after every reprimand, and worsened after rewards. Hence, the instructors held the opinion that students should not be rewarded, and that indeed they should be punished even for trivial reasons in order to improve their performance. Their belief, being in contradiction with psychological motivation theories, was in fact proven correct after an analysis of pilot performance and reinforcements’ data. However, what lay behind this finding (which contradicted motivational theories) was statistical regression: after an exceptionally bad flight it is likely that the next performance will be closer to the mean, that is, better; and after

an exceptionally good flight, the next one will likely be less surprising, that is, worse than the last. Statistically, this behavior would hold without instructor intervention. The instructors had discovered statistical regression, but had assigned it to the wrong cause, with unpleasant consequences.

- **Complexity of the decision environment:** Several factors can add up to the complexity of the environment: too much information, time pressure and distractions cause bad judgments and heavier reliance on dangerous heuristics.
- **Emotional stress:** Even when the source of emotional stress is unrelated to the present situation, it reduces the care with which people select and process information, and it may precipitate them to make panic judgments.
- **Social pressures:** People may try to please or confront other people. It is important to note that this is not a cognitive bias, its nature is motivational: it stems from our desire of recognition and acceptance, or from our fears.
- **Group think:** The phenomenon by which a group takes a decision which no (or almost no) group member would have taken individually.
- **Consistency of information sources:** This may be better stated with the phrase "If everyone says so, it must be true". Singer [Sin82] shows how some "mythical numbers", that is, factual numbers frequently repeated by authority figures and the media, are proven incorrect with even the simplest calculations.

3.1.3 – Output biases

The way that people are asked to express their judgment may itself be a source of bias, independently of the processing of information that led to their original judgment. Some examples of this type of bias are:

- **Scale effects:** The scale on which a person is asked to give his answer may affect the answer he gives. For example, people assign probabilities differently on a percentage scale than when *x:y* odds are used.
- **Illusion of control:** According to Hogarth, activities such as planning or forecasting induce feelings of control over the uncertain future. This is most evident in sports predictions and bets.

3.1.4 – Feedback biases

Without feedback to actions, learning is impossible. The natural cycle of actions and their feedback, however, is not always at reach: in some situations, feedback is unavailable, inconsistent, ambiguous or delayed. Thus we may relate causes with the wrong effects, or confuse randomness with determinism. Some specific examples of feedback bias are:

- **Overconfidence:** Oskamp [Osk82] affirms that practice or familiarity, when there is lack of proper feedback, causes people's confidence in their accuracy to increase, but their actual accuracy remains constant –or even worse, decreases.
- **Gambler's fallacy:** This is an inability to perceive randomness for what it is. After observing a sequence of coin tosses with "heads" outcomes people tend to believe that "tails" is a more likely outcome the next time.
- **Success/failure attributions:** Ross and Anderson [RA82] show how people are inaccurate at assessing the causes of their own successes and failures: we tend to attribute success to our own skill, and failure to chance or circumstances.
- **Logical fallacies in recall:** If people cannot recall details about an event they may produce a fictitious, but logical, reconstruction of its facts. This is known to happen with eyewitness testimonies.

- **Hindsight bias:** Also known as the “I knew it all along” bias. According to Fischhoff [Fis82], in retrospect people don’t seem to be surprised about some situation’s outcome and can easily produce arguments that explain such outcome convincingly, although before its occurrence they were quite uncertain of what would happen.

3.1.5 – Severity and quantity of biases

The previous lists and discussion may cause either amusement or frustration. It seems there are too many sorts of biases, and through them human judgment is seriously unreliable. Hogarth argues, however, that in the majority of situations these biases are harmless, and that perhaps they are the result of evolutionary tradeoffs in human cognition that help us to successfully save time and effort in most natural circumstances. It may be better to have problems differentiating between representativeness and probability, than to require a complex mental computation to correlate a representative example with the class it came from.

What should be concluded is not that human thought is fundamentally flawed, but that it relies on some heuristics and motivations that, while being generally beneficial, are the source of constant, repeating and predictable errors.

3.2 – Anchoring and Adjustment

Anchoring and adjustment is a phenomenon observed when people face choices under uncertainty, and is particularly notorious when the result of the choice can be expressed as a number within a range. If judgment of the matter is difficult we appear to grasp an *anchor*, that is, a tentative, even if unlikely, answer; and we *adjust* such answer up or down according to our intuition or experience to reach the final result.

The reason why anchoring and adjustment is considered a bias is that the adjustment humans commonly apply to the initial anchor is frequently insufficient to compensate for the negative effects of the anchor. Anchors, then, have the effect of attracting answers towards them and away from the correct number.

Tversky and Kahneman [TK82] first reported this phenomenon by describing the following experiment: Participants were individually presented a wheel of fortune with numbers from 0 to 100. The experimenter spun the wheel in front of the participant, and after it stopped –in a position evidently random- he questioned the participant to estimate various quantities, stated in percentages. For example, participants would be asked to give the percentage of African countries in the United Nations. Participants were first asked to indicate if the correct answer to the question was higher or lower than the random number that came up in the roulette, and then to estimate the correct value by moving upward or downward from the random number.

Tversky and Kahneman report that the arbitrary initial numbers obtained from the roulette had a marked effect on estimates: the median estimate for the African countries question was of 25 for people that received a 10 as their anchor, and 45 for those who received a 65. The researchers summarized the phenomenon as “different starting points yield different estimates, which are biased toward the initial values”.

Since then, the phenomenon has been studied thoroughly, and although the cognitive processes involved in it have not been singled out, the existence of the heuristic is now rarely questioned. It has been shown to happen in situations far more ordinary than the experiment described above, such as in general knowledge issues, probability estimates, legal judgment, pricing decisions and negotiation [MS01].

For example, [CB96] indicates that anchoring occurs in legal applications, and suggests that “plaintiffs would do well to request large compensation awards” to bias the awards granted by jurors. [NN87] demonstrated that professional real estate pricing decisions are also subject to anchoring biases, altering the pricing decisions of both experienced and inexperienced real estate professionals (although with a stronger impact on inexperienced professionals).

Initial anchors do not even need to be recognized as starting points for the solution. [AWA02], for example, affirms that the duration of a criminal sentence partially depends on numbers that are fresh in the mind of the sentencing judge. However, [MS01] reports that semantic anchoring effects are more potent than purely numeric effects; that is, the anchor is more effective if it is regarded as a possible, meaningful solution to the problem at hand.

A series of experiments by Wilson et al. [WHB93] provide interesting insights on the anchoring and adjustment phenomenon. Their results indicate that (a) anchoring occurs if people pay sufficient attention to the anchor value, (b) that knowledgeable people are less susceptible to anchoring effects, and (c) that anchoring appears to operate unintentionally –it is difficult to avoid even when people are forewarned.

3.3 – Judgmental biases in software estimation

Software estimation is frequently approached as a technical problem with clear specifications and a correct answer. This is probably the result of designing models and techniques with data from projects previously developed, for whom there is a set of metrics and all problems and risks eventually surfaced and were recorded. Applying such models to future projects involves an amount of foresightedness that is highly unrealistic, and therefore judgment has to be made under uncertainty.

Although most research in software estimation is technical and concerned with refining models or adapting them to new lifecycles and development dynamics, there has been a growing field within software estimation that attempts to discern predictable elements of human behaviour from software estimation processes.

The following studies have contributed in attracting attention to software estimation as a primordially human activity, deeply related to thought processes.

On the topic of estimator overconfidence, [JTM04] describes an empirical study which found confidence of estimators in their own estimates was unjustifiably high. Furthermore, estimators do not appear to handle well different estimation confidence percentages. No distinction seems to be made between 50%, 75%, 90% and 99% confidence in an estimate.

On estimator experience, Hill et al. [HTA00] affirm that "a study of six software project leaders' estimates over a period of three years showed no significant learning effect". These results may be explained by a lack of a proper feedback loop to the thought processes involved in software estimation, or by the continuously changing environment in which software projects take place.

Two studies address the issue of expectations in estimates, and one even raises a link between anchoring and adjustment and estimation [JS01], even if the type of estimation the study is concerned with is not related to software, but to student coursework. According to this empirical study, when asked to perform a work breakdown structure (WBS) of activities to be performed for an undergraduate course, and if given a low or a high anchor, students will correspondingly produce unrealistically low or high WBS estimates.

The second study has several similarities to the one described in this thesis. [JS04b] reports an empirical study where customer expectations were directly stated to estimators of a short software task. Participants were instructed to estimate the task using a WBS analysis. These expectations were found to cause an impact in the final estimates.

Another study explores performance evaluations that managers assign to estimators [Jør04]. Managers have been found to prefer estimators that produce narrow (and wrong) estimates over those who produce wide (and correct) estimates. Subjectively, they seem to believe that estimators that give narrow answers are more knowledgeable than their complements, and if they are wrong it may be because they had a run of bad luck.

Finally, an interesting paper by Abdel-Hamid and Madnick [AM86] reports that estimates are likely to be a factor in the real effort of projects. That is, estimates may be self-fulfilling prophecies, where low estimates are matched with short projects by compromising and eliminating unnecessary features, and high estimates are matched with long projects that take a more detailed approach, gold-plate features and tolerate a greater amount of requirements creep.

In summary, these results taken together lead to the conclusion that software estimation is a field that benefits from being studied with a psychological perspective, and they call for further efforts in this direction.

Chapter 4

Research Questions

The underlying assumption of this study is that software estimation is essentially a human judgment activity, and that as such it is subject to judgmental biases. The previous discussion on the nature of software estimation supports this view, and efforts to automate estimation, although successful in giving shape to such a freeform activity, do not eliminate or reduce the intervention of human judgment.

Software estimation, being subject to judgmental biases, is a prime candidate to suffer the effects of anchoring and adjustment. The main reasons are:

- **Judgment under uncertainty:** The complexity of software development, the magnitude of factors that can speed it up or down, and the enormous variation in impact most of these factors have, collaborate in the uncertainty involved in estimation. Documented failures of software estimates, mainly in the form of schedule overruns [Sta94], stand as witnesses to the difficulty of accurately predicting the amount of effort required in software projects. When faced with an estimation task, estimators need to handle uncertainty and ambiguity in the form of vague and missing requirements, assessments on the experience and skill of developers, rigour of non-functional requirements, customer expectations, and a whole range of unforeseeable outside events that can alter the time and effort needed to finish the project. It has been demonstrated [KST82] that heuristics and biases occur when humans are faced with such uncertainty; therefore, it is natural to expect them to occur in software estimation activities.

- **Quantitative estimates:** A specific requirement for the anchoring and adjustment bias is that the answer that subjects must provide should be in the form of a number, not of a subjective statement or a binary decision. Software effort estimates, commonly stated with units such as man-months, weeks, months or dollars, fulfill this requirement.
- **Natural use of anchors among managers and developers:** In the software development field anchors are produced and communicated within the development team and customers almost unconsciously. Phrases like "Do you think you'll finish by mid February?" are common and expected in the software industry –there is no concern for the effect that such questions may have upon the accuracy of the response.
- **Lack of solid framework for software development:** Although there are several efforts to standardize the software development process with a considerable number of adherents [Hum89], it is still largely an immature and unexplored discipline. New and sensible methodologies are proposed all the time, but a clearly superior technique has not yet been found –nor, for that matter, has it been proven that there *may be* a technique that would be superior to any other for all types of software projects. This lack of a standard framework for software development increases the uncertainty of estimation and makes it difficult to explore in detail the consequences of changing methods and practices in a development team.

The following questions were formulated to help identify whether anchoring and adjustment is indeed an effect worth considering while estimating software projects.

4.1 – Existence of anchoring and adjustment effect

We should first be concerned with defining if an anchoring and adjustment effect is observed in an empirical study. The first research question is therefore: Does the phenomenon of anchoring and adjustment influence software estimation processes?

To answer this question the empirical study should be designed in such a way that a variation in anchors is the only relevant difference among estimators. The following chapters describe the study that was designed and executed to address this issue.

Although this is the main research question of the thesis, there are three other significant topics worth exploring. The answer to them stems from the results of this question.

4.2 – Variation in the effect between experienced and inexperienced estimator subgroups

It has previously been observed that anchoring and adjustment effects are weaker for knowledgeable, or expert, participants [NN87]. However, within software estimation research, it has been found that estimation experience is a bad indicator of estimation accuracy, that is, experience does not seem to lead to reliability within software estimation [JS04]. This combination of results makes it hard to predict whether experienced software estimators will be as influenced by anchors as their inexperienced colleagues.

For these reasons, our second research question is: Is the influence of anchoring and adjustment weaker for estimators that have had previous experience estimating software projects?

4.3 – Variation in the effect between users of model-based techniques and expert-based techniques

Since research in model-based techniques and expert-based techniques has not yet provided strong evidence of the superiority of either type of technique [Jør04] it would be interesting to see if either method provides an advantage over the other as far as anchoring and adjustment effects are involved. Our third research question can be formulated as follows: Is the influence of anchoring and adjustment stronger for estimators that rely solely on expert-based estimation, as opposed to estimators that use a model-based technique?

4.4 – Compensation of anchoring effects by confidence ranges

It is unrealistic to expect that an estimate expressed as a number will have much accuracy. Ranges of estimates, either expressed as an interval or as a central point with a confidence range in percentage, help to frame an estimate within a timescale and give an opportunity to estimators to express the amount of uncertainty they may have in their own answers.

If estimators are allowed to express their estimates with a confidence range, they should be able to compensate for the ambiguity and judgmental biases inherent in estimation processes.

The fourth question we are concerned with is therefore: Does the confidence (or lack thereof) estimators have in their answers compensate for possible anchoring and adjustment biases?

This question is relevant because the effect of anchors could be irrelevant if estimators are realistic about their own performance and give wide confidence ranges. However, if this is not the case, even giving estimators the advantage of stating their results with intervals may not be enough to counteract the judgmental biases involved in software estimation, and other steps should be taken to compensate for them.

Chapter 5

Experiment Plan and Design

An empirical study in software estimation was designed to answer the research questions posed in the previous chapter. This chapter contains the description of the experiment, its design, variables, hypotheses and threats to its validity.

5.1 – Experiment Design

The experiment consisted of a software estimation exercise that consenting participants worked at individually. They were given the problem of estimating how long they think it would take to deliver a specific software application.

The application was described in a ten-page document named "Software Requirements Initial Report" (see Appendix 3). The client for the application lies within a hypothetical foreign trade agency, and the main task of the application was to process, analyze and report statistics on foreign trade in the area the agency deals with. Two reasons were relevant to choose this domain:

- **Lack of familiarity of any participant with domain:** Having a percentage of participants familiar with the domain would give them an advantage over other participants and would bias the experiment. For this reason, foreign trade statistics analysis was preferred over more common domains such as e-commerce applications, CRM or ERP tools.

- **Previous experiment designer experience:** The designer of the experiment had previously worked in a project similar in its domain, but different in its scope, and could point to peculiar requirements in the domain and reasonable choices for anchor values (see below).

The format for this document was adapted from the IEEE Recommended Practice for Software Requirements Specifications [IEEE98].

The requirements report was complemented by another document, named "Project Setting" (see Appendix 4). This document, three pages long, discussed particularities of the client organization and of the development team in charge to develop the application. It included information such as experience of the developers, insights into their team dynamics, expectations of future users of the system and some subtle, non-functional requirements. This document, along with the requirements report, was the only information participants were given on the application domain and its environment.

Participants worked on this problem individually. They could take as much time as they desired, and although their performance was not timed, the majority of them reported taking from one to two hours in the exercise.

All participants had total freedom on their choice of estimation techniques, as long as they worked on the exercise by themselves. They could use software estimation tools to aid their judgment if they desired.

Once they finished their estimation they needed to answer a questionnaire (see Appendixes 2 and 5). The most relevant questions in it were:

- Give your estimate for the duration of the project described in the attached documentation, in months, to the nearest integer
- I think that if this project was really developed, my estimate might be off by as much as ___%

- Justify your estimation. Try to justify it in such a way that a reader may understand and follow your reasoning and probably reach the same conclusion.

That is, they were required to submit their estimate, a confidence range, and a justification for their answers.

The rest of the questions in the questionnaire were included in case they would give additional insights to the estimation process and to assess the quality of the experiment documentation. They were:

- I think the estimation I performed was… (answer was given in a scale from 1 to 7, 1 being "very unreliable" and 7 "very reliable")
- My previous estimation experience includes (check all that apply). (Options included involvement in estimation of medium to large projects, estimation of small projects, courses that had estimation as a topic, witnessing software estimation processes, and self-learning)
- I felt the documentation was… (answer was given in a scale from 1 to 7, 1 being "very uninformative" and 7 "very informative")
- Explain your strategy to estimate software development projects

The estimation experience question was the only means by which estimator experience was assessed in this study. There were no attempts to probe this self-assessment, nor any definitions of what does each participant mean by, for example, "medium to large projects".

All participants signed a consent form and were paid $10 as a token of gratitude for their involvement in the study.

There were three different conditions in the experiment. Each participant was assigned to one condition, with the intention of having each condition a similar proportion of participation and experienced subjects as the others. The only difference among the three conditions was a paragraph in the second page of the Project Setting document. In a box with quotes from a middle manager in the client organization, one of the sentences was altered in each group.

For the experiment's control condition, the manager was quoted as saying:

"I'd like to give an estimate for this project myself, but I admit I have no experience estimating. We'll wait for your calculations for an estimate."

For a second, "2-months" condition, the quote was modified to include an anchor. It read as follows (emphasis added here):

"I admit I have no experience with software projects, but I guess this will take about ***2 months*** *to finish. I may be wrong of course, we'll wait for your calculations for a better estimate."*

Finally, a third, "20-months" condition, had a high anchor in the manager's statement. The statement was (emphasis added):

"I admit I have no experience with software projects, but I guess this will take about ***20 months*** *to finish. I may be wrong of course, we'll wait for your calculations for a better estimate."*

All other data were identical among conditions.

There are several issues worth noting at this point:

- The difference among anchors is an order of magnitude. This difference is quite large, but sensible. According to [BCH+95], reasonable estimates on the very first stages of project development may differ with a proportion of as much as 16:1.
- The anchor given to participants is semantically linked to the answer participants are asked to provide. This is relevant since, as Mussweiler and Strack note [MS01], semantic anchors

are more effective than simple numeric anchors. Furthermore, a numeric anchor without a semantic link to the estimate was not feasible in this experiment since there are several numbers in the documentation (performance goals and years of experience, for example) that could be taken as numeric anchors as well.

- The quoted individual is not pushing his guess as a starting point for negotiation. He acknowledges his own lack of experience in estimation and labels his number as a guess.
- Participants did not hear the individual saying this sentence, they read about it. For this reason, they may be less likely to attempt to please him by giving a number close to the anchor. Attempting to please is also a judgmental bias, but of a social, not cognitive, nature. The research questions of this study are cognitively oriented, and therefore it is important to limit the influence of social biases in its design.

5.2 – Variables

A formal breakdown of the variables recorded for this experiment should be described. All of the following variables were monitored:

5.2.1 – Independent variables

Only one independent variable was used: the anchoring statement discussed in the previous section. As was mentioned, it could take three values, named “2 months”, “20 months” and “control” conditions.

5.2.2 – Controlled variables

In addition to the independent variable, while assigning participants to each condition an initial assessment of their experience was obtained, in order to reach a balance of experienced participants in all conditions. Estimating experience could take three values in this study: (a) Experience in estimation of medium to large software projects; (b) experience in estimation of small software projects; and (c) only academic experience (through coursework or self learning). As has been previously discussed, estimator experience was determined by each participant, and their definitions of project size, involvement in estimation and amount of time dedicated to self learning, for example, were not explored.

5.2.3 – Dependent variables

Three dependent variables were considered of high importance for this study. They were:

- **Estimate:** This is the actual estimate as given by participants. It can only take the form of a positive integer representing the number of months that the estimator considers as the most likely possible duration for the project. Participants were asked to round their estimates to the nearest integer; no other types of numbers were received.
- **Confidence Range:** Expressed as a percentage that can be added or subtracted from an estimate to reach an acceptable confidence range. For example, a 10 months estimate with 30% confidence range would consider all points between 7 and 13 months as a probable duration for the software project.
- **Estimation Method:** Participants were not asked to name the estimation method they used. However, they were asked to provide a justification for their estimate. These justifications were analyzed to classify the estimation technique in one of two general subgroups: Model-based and expert-based. Further classifications within each subgroup were LOC-based or FP-based (for model techniques) and WBS (work breakdown structure) or intractable

process (for expert techniques). If a participant used more than one technique, an assessment of what was the main technique used was required –each participant was considered to have used only one primary technique, even though the use of secondary techniques changed the original estimate.

The questionnaire that participants answered included other questions that required additional monitoring. These answers are represented by the following variables:

- Subjective reliability of estimate (in a number from 1 to 7)
- Quality of the information in the documentation (in a number from 1 to 7)
- General strategy for estimating software projects (unrestricted text)

5.3 – Hypotheses

Hypotheses for the study were formalized. The following are the general null hypotheses for the experiment:

- $H_{0,\ LOW\text{-}HIGH}$: Estimates of participants given a low (“2-months”) anchor are not statistically different from estimates of participants given a high (“20-months”) anchor.
- $H_{0,\ LOW\text{-}CONTROL}$: Estimates of participants given a low (“2-months”) anchor are not statistically different from estimates of participants given no anchor at all.
- $H_{0,\ HIGH\text{-}CONTROL}$: Estimates of participants given a high (“20-months”) anchor are not statistically different from estimates of participants given no anchor at all.

Each null hypothesis has an alternate hypothesis rejecting it.

A similar set of hypotheses was generated for analyzing the results from experienced participants (considering as “experience” the self-assessed involvement in estimation processes of software projects of any size). The following are the null hypotheses for such participants; alternate hypotheses rejecting them were also generated but are not included here:

- $H_{0,\ LOW\text{-}HIGH,\ Experienced}$: Estimates of experienced participants given a low (“2-months”) anchor are not statistically different from estimates of experienced participants given a high (“20-months”) anchor.
- $H_{0,\ LOW\text{-}CONTROL,\ Experienced}$: Estimates of experienced participants given a low (“2-months”) anchor are not statistically different from estimates of experienced participants given no anchor at all.
- $H_{0,\ HIGH\text{-}CONTROL,\ Experienced}$: Estimates of experienced participants given a high (“20-months”) anchor are not statistically different from estimates of experienced participants given no anchor at all.

Two more sets of hypotheses were generated for the subgroups of estimators that used primarily a model-based technique and those who used primarily an expert-based technique. These hypotheses are similar in structure to the previous. For the model-based techniques users the null hypotheses are:

- $H_{0,\ LOW\text{-}HIGH,\ Model\text{-}Based}$: Estimates of participants who used primarily a model-based estimation technique and were given a low (“2-months”) anchor are not statistically different from estimates of participants who also used a model-based estimation technique and were given a high (“20-months”) anchor.
- $H_{0,\ LOW\text{-}CONTROL,\ Model\text{-}Based}$: Estimates of participants who used primarily a model-based estimation technique and were given a low (“2-months”) anchor are not statistically different from estimates of participants who also used a model-based estimation technique and were given no anchor at all.
- $H_{0,\ HIGH\text{-}CONTROL,\ Model\text{-}Based}$: Estimates of participants who used primarily a model-based estimation technique and were given a high (“20-months”) anchor are not statistically

different from estimates of participants who also used a model-based estimation technique given no anchor at all.

And for the expert-based techniques users, these are the relevant null hypotheses:

- $H_{0,\ LOW\text{-}HIGH,\ Expert\text{-}Based}$: Estimates of participants who used primarily an expert-based estimation technique and were given a low ("2-months") anchor are not statistically different from estimates of participants who also used an expert-based estimation technique and were given a high ("20-months") anchor.
- $H_{0,\ LOW\text{-}CONTROL,\ Expert\text{-}Based}$: Estimates of participants who used primarily an expert-based estimation technique and were given a low ("2-months") anchor are not statistically different from estimates of participants who also used an expert-based estimation technique and were given no anchor at all.
- $H_{0,\ HIGH\text{-}CONTROL,\ Expert\text{-}Based}$: Estimates of participants who used primarily an expert-based estimation technique and were given a high ("20-months") anchor are not statistically different from estimates of participants who also used an expert-based estimation technique and were given no anchor at all.

To address the fourth research question, dealing with the confidence of estimators in their own answers, an additional set of null hypotheses was generated:

- $H_{0,\ MaxLow\text{-}MinHigh}$: The maximum estimates of participants given a low ("2-months") anchor are not statistically different from the minimum estimates of participants given a high ("20-months") anchor.
- $H_{0,\ MaxLow\text{-}CONTROL}$: The maximum estimates of participants given a low ("2-months") anchor are not statistically different from the base estimates of participants given no anchor at all.

- $H_{0,\ MinHigh\text{-}CONTROL}$: The minimum estimates of participants given a high (“20-months”) anchor are not statistically different from the base estimates of participants given no anchor at all.

For all these sets of null hypotheses, alternative positive hypotheses rejecting the originals were also generated.

5.4 – Threats to Validity

It is important to explore the threats to the validity of this experiment. The following discussion on threats is based on the list proposed by Wohlin et al. [WRH+00].

5.4.1 – Conclusion validity

The group of participants that performed the software estimation exercise was relatively heterogeneous, consisting of Computer Science graduate students and software industry professionals. A wide diversity in background and experience among participants would make it difficult for conclusions of this study to be applicable to the specific group of software estimators. Another aspect of this threat is the fact that some participants had only learned about software estimation through coursework and self-learning, and had never been required to produce an estimate in a real-world software project.

However, it should be considered that all of the participants had at least the basic qualifications to be required to perform real software estimation; that is, they all were potential software estimators with enough authority, either because of background, academic formation or a combination of both, to produce estimates in real software development projects. For this reason they can be regarded as part of the same group, and the influence of this threat is reduced.

5.4.2 – Internal validity

Participants to this study were volunteers who responded to an invitation. According to [WRH+00], volunteers are specially motivated, and are therefore not representative of the whole population. This was a necessary evil, since the alternative of hiring a significant number of professional software estimators and paying them their usual fees for their services was not economically feasible for this study.

5.4.3 – Construct validity

A flaw of this experiment is that it uses only one set of project documents, that is, it may be suffering from a mono-operation bias. It would have been interesting to perform it with several (at least two) different software projects and see if the relations between anchors and control groups are replicated among them. Economical limits and a difficulty to find willing participants prevented the study from going in that direction.

Another threat for construct validity was the set of expectations that the experimenter had on the outcome of the study. Planning and designing the study was done by a single person, and an unconscious bias in favour of those expectations may have taken place. Although steps were taken to avoid this possibility, the only way to truly neutralize it would have been the involvement of additional experimenters without expectations in the outcome of the study, which was not feasible in the environment it took place.

5.4.4 – External validity

An external validity threat for this experiment consists of an interaction of selection and treatment [REF: Wohlin in RE2]; that is, the subject population may not be representative of the population to which the results should be generalized. Not all participants had been asked to perform an estimation task before –therefore, generalizing the results to include software estimators is questionable. However, the threat is reduced when we consider that every participant has the potential and basic qualifications, either because of industry background or academic formation, to be a competent software estimator.

Another threat to external validity lies in the fact that real software estimation carries consequences that may be felt for a long time by the people involved, potentially altering their career paths. Participants in this study knew they would not be held accountable for their answers, and this difference between the experiment and real estimation experiences may affect the results. This is a consequence of performing a controlled experiment. The alternative would be to observe real software estimations within their natural environments, but meaningful statistical observations could hardly be drawn from such efforts.

Chapter 6

Experiment Execution

The experiment described in the previous chapter was executed during the second half of the year 2004. Many participants were recruited through e-mail invitations sent to graduate student mailing lists. Personal and indirect contacts helped to recruit the rest of the participants.

After candidate participants expressed interest in the experiment, they were visited at the time and place of their choice. There they were given their set of experiment documents and instructions. Basic descriptions of each document and a summary of the instructions were also stated orally.

Participants were allowed to work on the exercise as long as they wanted. They could use software tools and reference books if they wished. They were asked to contact the experimenter back when they had finished the exercise. Most of them reported having finished before three days had elapsed since they were visited. An informal average of the time needed to finish the exercise is of about one hour and a half, with some participants taking close to three hours and some finishing in a little less than an hour.

Participants were not told the purpose of the study. They were told they were participating in a “software estimation experiment” without going into further detail. All participants signed a consent form and were guaranteed anonymity.

There were 23 participants in this study (plus one participant whose answers had to be discarded because they were incomplete and the participant did not intend to collaborate further). The majority of them (78%) were graduate students in Computer Science, the remaining 22% were professionals from the software industry. 57% of participants declared they had been involved in real software estimation activities before (22% were involved in medium to large software projects, 35% only in small projects). 43% had only academic experience in the area.

An even distribution among all conditions was intended. The final number of respondents, however, was variable among conditions due to participation cancellations. The “2-months” condition received 9 responses. The control condition, 6 responses; and the “20-months” condition, 8 responses.

Each participant’s answers were separated from their names and recorded. The estimates among groups of participants were analyzed using independent *t*-tests for each hypothesis.

Chapter 7

Data Analysis and Interpretation of Results

This chapter presents all results obtained from the study on software estimation. Some general results will be shown first, to be followed by a deeper discussion and interpretation of the most meaningful results.

7.1 – General Results

Responses obtained by this experiment presented a very wide range of estimates. The shortest estimate to deliver the software project was of 3 months; the longest, 28 months. This gives a proportion of 9.33:1 from longest to shortest. The average estimate was 10.9 months.

When we consider the confidence limits given by estimators, the range of estimates widens significantly. The minimum length for the project considered by any single estimator was 2 months; the maximum, 44.8 months. This gives a proportion of 22.4:1 from the longest to the shortest, which is almost perplexing.

Participants allowed themselves an average confidence range of +/-26% on their base estimate. The narrowest confidence range was of 10%; the widest, 60%. Incidentally, according to [Boe81], and considering the variety in estimates received, the latter is more sensible than the former, at least on the initial stages of a software development lifecycle.

As mentioned in Chapter 5, estimation techniques were classified in two groups: Model-based estimation and expert-based estimation. Model-based was further subdivided into LOC-based estimates and FP-based estimates; expert-based was subdivided into WBS (work breakdown structure) analyses and freeform processes.

Using this classification scheme, the choices of estimating techniques in this study were as follows: 31% of estimators chose a primarily model-based approach; 69% an expert-based approach. Of the whole pool of participants, 22% used a LOC-based technique, 9% a FP-based technique, 39% a WBS analysis and 30% a freeform process.

These results fall in line with previous surveys on use of estimation techniques, although the use of model-based techniques was slightly higher than expected. Other types, such as learning-based techniques, were not feasible to use in this experiment's setting.

7.2 – General Anchoring and Adjustment Results

Perhaps the best way to illustrate the general results from the experiment is through a graph. Figure 7.1 shows the results of the exercise.

The chart is divided in three areas. The lower area corresponds to participants in the "2-months" anchor. The middle area shows the responses in the control condition. Finally, the upper area corresponds to the "20-months" condition. For each condition the mean value is illustrated with a vertical bar. For the "2-months" and "20-months" conditions, the corresponding 2-months and 20-months lines are also shown. Each estimate shape shows the base estimate as a small diamond and the confidence range to the left and right of the base.

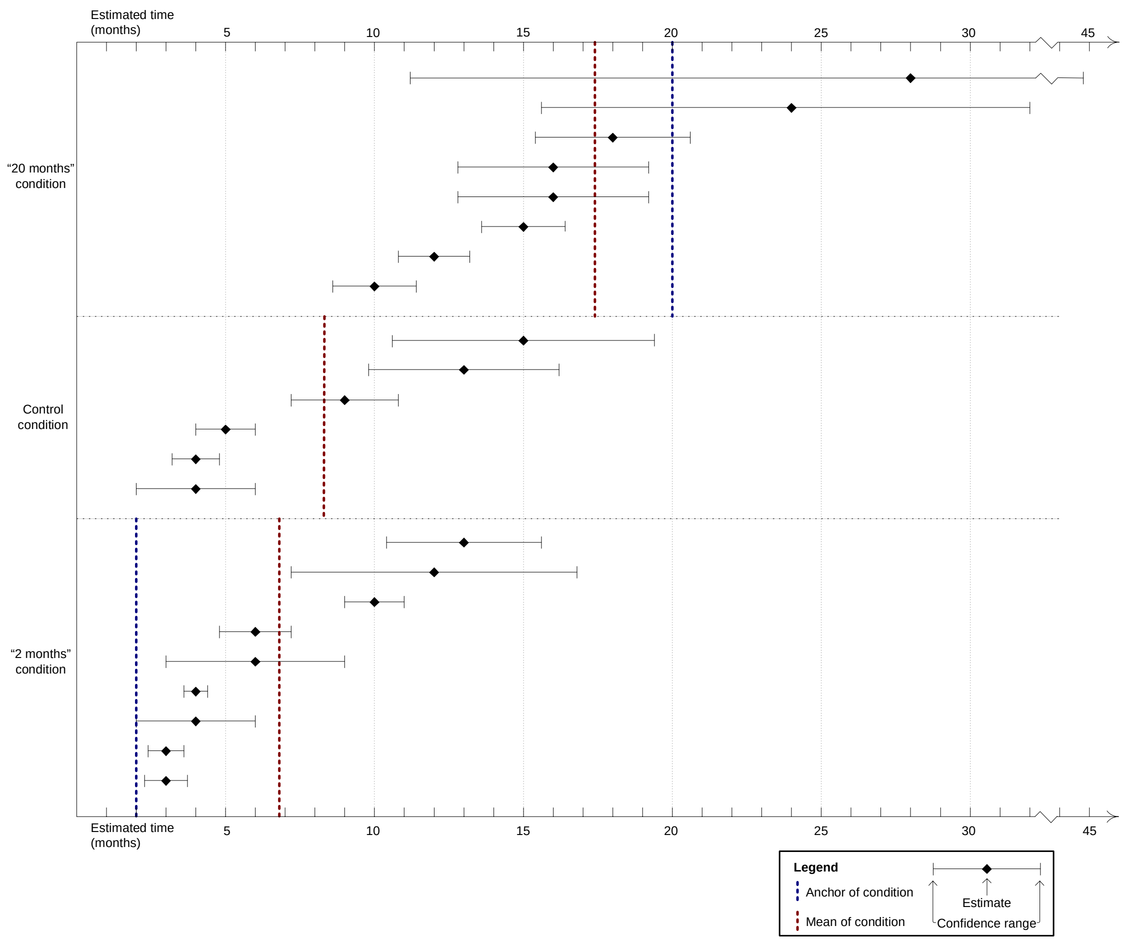

Figure 7.1 – All estimators data

Although the patterns on each condition are visibly significant, the following numbers help to give numerical clarity to the chart. The “2-months” condition participants had a mean estimate of 6.8 months. Their median estimate is 6 months, and their standard deviation 3.7.

The control condition has a slightly higher mean estimate, at 8.3 months; a median of 7 months and a standard deviation of 4.4.

The “20-months” condition’s mean estimate is 17.4 months; its median is 16 months and the standard deviation 5.6.

Within each group there is a high variation of results as well. The “2-months” condition’s greatest estimate is 4.33 times greater than its lowest. The corresponding proportion is 3.75 for the control condition and 2.80 for the “20-months” condition.

We should now consider how these data reflect the hypotheses established for the experiment. There are three hypotheses significant at this point; those that refer to the basic effects of anchoring and adjustment in software estimation: $H_{0, LOW\text{-}HIGH}$, $H_{0, LOW\text{-}CONTROL}$ and $H_{0, HIGH\text{-}CONTROL}$.

The first null hypothesis, $H_{0, LOW\text{-}HIGH}$, states that there is no statistical difference between the “2-months” and the “20-months” conditions. The *t*-test for this hypothesis gives $t = 4.273$, and the null hypothesis is rejected ($p < 0.001$).

The second null hypothesis, $H_{0, LOW\text{-}CONTROL}$, says that there is no statistical difference between the “2-months” and the control conditions. The *t*-test gives $t = 0.661$, the null hypothesis cannot be rejected ($p > 0.1$).

The third null hypothesis, $H_{0, HIGH\text{-}CONTROL}$, states that there is no statistical difference between the “20-months” and the control conditions. The *t*-test gives $t = 3.137$, and the null hypothesis is rejected ($p < 0.01$).

That is, the “2-months” and the control conditions were found to be statistically different than the “20-months” condition. However, a significant difference between the “2-months” and the control conditions was not found.

Therefore, the anchoring and adjustment heuristic was found to take place in software estimation processes, at least when the effects of providing a low anchor are compared with the effects of providing a high anchor ($p < 0.001$) and when the effects of providing no anchor are compared with the effects of providing a high anchor ($p < 0.01$). The effects of anchoring and adjustment between a low anchor and no anchor at all are suggested by the results, but were not found to be statistically significant. A discussion in the next chapter suggest several reasons why this may be so.

7.3 – Experienced Participants Results

We will now filter the results of the experiment to include only those participants who declared to have real-life experience estimating software projects; which is 57% of the total pool of participants. Figure 7.2 displays the results for this subset of participants.

As can be seen in the chart, the pattern remains unchanged after removing inexperienced estimators, although the results are slightly weaker due to the reduced number of participants.

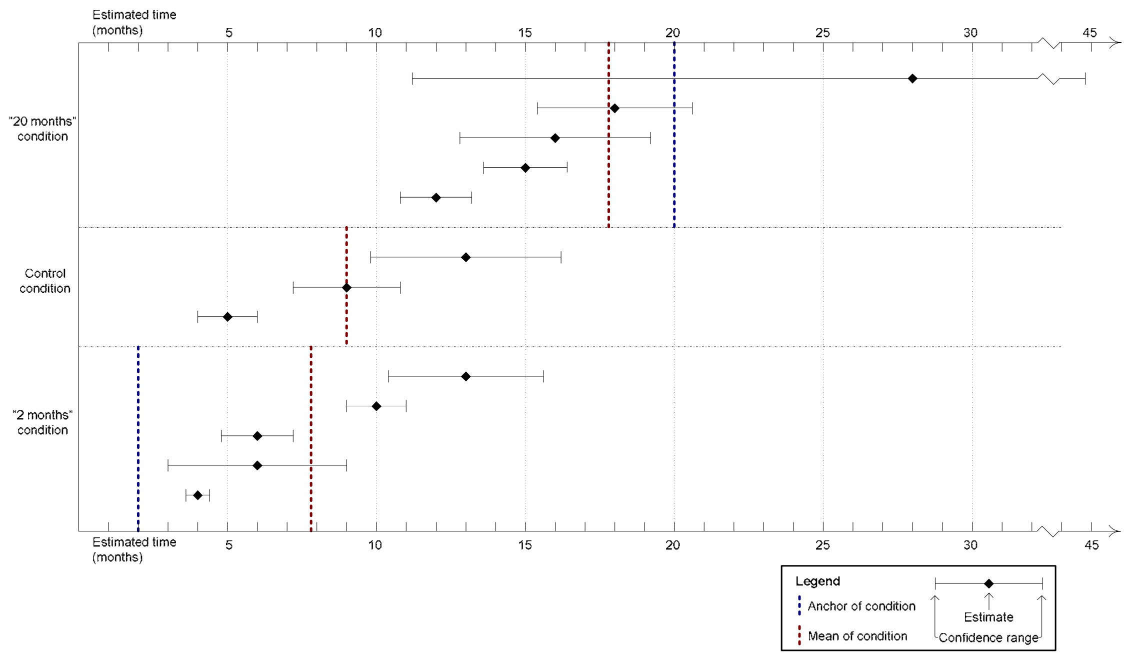

Figure 7.2 – Experienced estimators data

Within this subgroup of experienced estimators, the “2-months” condition mean estimate is 7.8 months, the median 6 months and the standard deviation 3.2. The corresponding numbers for the control condition are a mean of 9.0 months, median of 9 months and standard deviation 3.3. Finally, for the “20-months” condition, the mean is 17.8 months, the median 16 months and the standard deviation 5.5.

Considering the three relevant null hypotheses for experienced estimators we have the following results:

The first null hypothesis, $H_{0,\ LOW\text{-}HIGH,\ Experienced}$, gives a *t*-test of $t = 3.150$, and the null hypothesis is rejected ($p < 0.02$).

The second null hypothesis, $H_{0,\ LOW\text{-}CONTROL,\ Experienced}$, has $t = 0.425$, the null hypothesis cannot be rejected ($p > 0.1$).

The third null hypothesis, $H_{0,\ HIGH\text{-}CONTROL,\ Experienced}$, gives $t = 2.462$, and the null hypothesis is rejected ($p < 0.05$).

What these results state is that all effects experienced by the generality of participants were also suffered by experienced estimators in particular.

7.4 – Expert-based Techniques Results

To address the hypotheses generated for expert-based techniques users, we filtered the data to include only those participants that used such techniques primarily. Their results are shown in Figure 7.3.

Although the general pattern maintains in this subgroup of participants, there are two differences between them and the previous two groups. The first is that the averages are lower within this subgroup than in the complete pool of participants. The second, the standard deviations are also lower, that is, each estimate is closer to the mean than in the two previous groups.

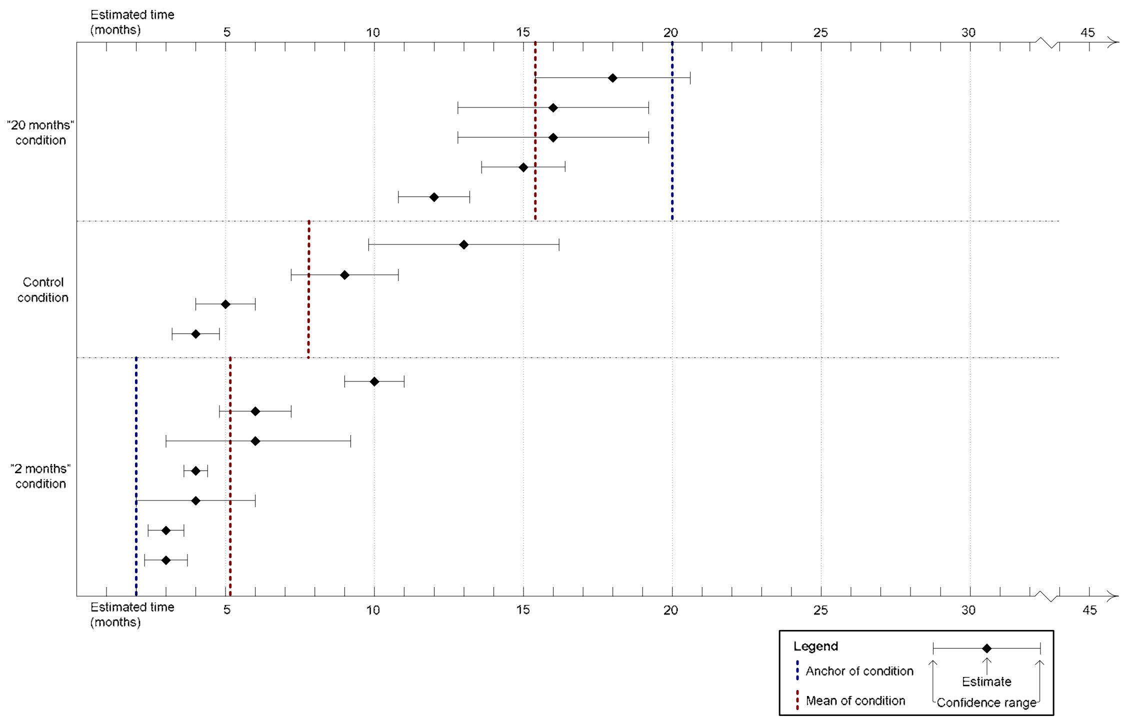

Figure 7.3 – Expert-based estimators data

The particular numbers are as follows. For the “2-months” condition, the mean is 5.1 months, the median 4 months and the standard deviation 2.3. For the control condition, the mean is 7.8 months, the median 7 months and the standard deviation 3.6. For the “20-months” condition, the mean is 15.4 months, the median 16 months and the standard deviation 2.0.

Applying independent *t*-tests for each of the three relevant null hypotheses for expert-based estimations results in the following:

The first null hypothesis, $H_{0,\ LOW\text{-}HIGH,\ Expert\text{-}Based}$, gives a *t*-test of $t = 7.567$, and the null hypothesis is rejected ($p < 0.001$).

The second null hypothesis, $H_{0,\ LOW\text{-}CONTROL,\ Expert\text{-}Based}$, has $t = 1.154$, the null hypothesis cannot be rejected ($p > 0.1$).

The third null hypothesis, $H_{0,\ HIGH\text{-}CONTROL,\ Expert\text{-}Based}$, gives $t = 3.358$, and the null hypothesis is rejected ($p < 0.02$).

Again, these results show that the effects of anchoring and adjustment are felt by estimators who choose an expert-based approach. In fact, these results were among the most powerful of the whole experiment. However, an effect comparing low anchor estimates and no anchor estimates was not found in this subset either.

7.5 – Model-based Techniques Results

The complement of the previous subgroup is that of estimators who used primarily a model-based technique to reach their results. Figure 7.4 shows their data.

There are not enough data points to reach conclusions for model-based estimators. Only 7 estimators chose to use models to solve the exercise, and the sample was not statistically significant to reach clear results.

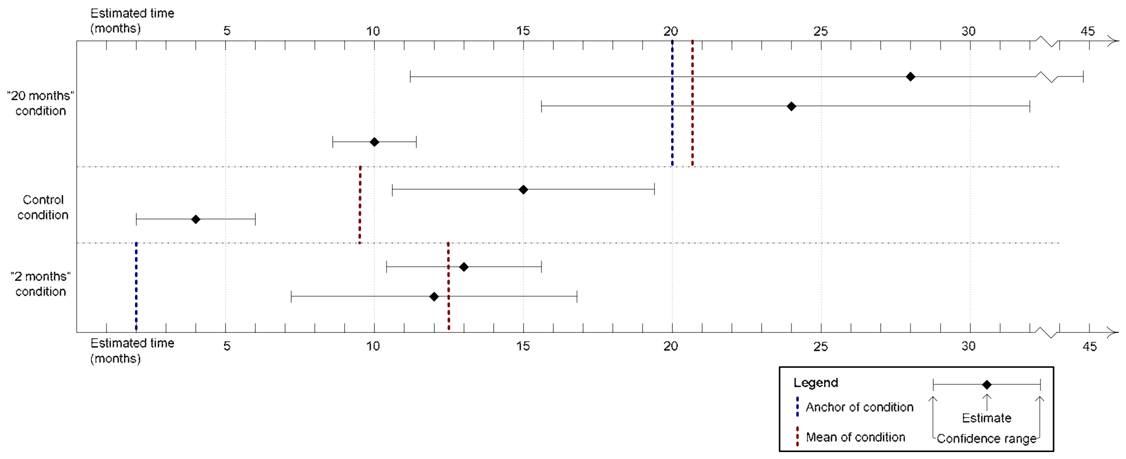

Figure 7.4 – Model-based estimators data

An analysis of Figure 7.4 shows that the same patterns visible on previous subgroups start to appear here. The "2-months" results are noticeably lower than the "20-months" results, and the control condition results are also lower than the "20-months" condition's. An anomaly here is that the control condition results are lower than the “2-months” condition estimates, which may be due to the low number of model-based estimates received.

The numbers for this subgroup are as follows. The “2-months” condition has a mean of 12.5 months, median of 12.5 months and standard deviation of 0.5. The control condition’s mean and median is 9.5 months, and its standard deviation is 5.5. The “20-months” condition has a mean of 20.7 months, a median of 24 months and a standard deviation of 7.7.

The three null hypotheses concerning model-based estimators could not be rejected with independent *t*-tests ($p > 0.1$ in all cases). It is impossible to predict if this was due to the low number of participants choosing model-based techniques or due to a weaker influence of anchoring and adjustment effects on this subgroup. Although existing data seems to indicate the former, it is not conclusive enough.

7.6 – Maximum-Minimum Results

There is one more set of hypotheses that need to be addressed, concerning the confidence that estimators have in their own answers. If we consider the total pool of participants (see Figure 7.1), but we focus our analysis on the maximum estimate considered by estimators in the “2-months” condition and on the minimum estimate considered by estimators in the “20-months” condition, do the differences in estimates still maintain a statistical significance? That is, are low anchor estimates’ worst-case scenarios significantly lower than high anchor estimates’ best-case scenarios?

The new numbers are as follows: The maximum estimates on the “2-months” condition have a mean of 8.7 months, median of 7 months and standard deviation of 4.8. The minimum estimates on the “20-months” condition have a mean of 12.8 months, a median of 13 months and a standard deviation of 2.2.

Applying independent *t*-tests with these data points for each of the three relevant null hypotheses gives the following results:

The first null hypothesis, $H_{0,\ MaxLow\text{-}MinHigh}$, gives a *t*-test of $t = 2.182$, and the null hypothesis is rejected ($p < 0.05$).

The second null hypothesis, $H_{0,\ MaxLow\text{-}CONTROL}$, has $t = 0.129$, the null hypothesis cannot be rejected ($p > 0.1$).

The third null hypothesis, $H_{0,\ MinHigh\text{-}CONTROL}$, gives a *t*-test of $t = 2.079$, and the null hypothesis is rejected ($p < 0.1$).

Therefore, these results show that the effect of anchoring and adjustment heuristics in software estimation can be so high that giving estimators the opportunity of including a confidence range in their estimates does not compensate for the bias suffered by this heuristic.

7.7 – Estimate Ranges Results

Although all hypotheses for this experiment have been explored, additional insights are found if the data from each condition are concentrated to show the general agreement that estimators may have between each other. Figure 7.5 displays this information.

For each condition, the chart shows the percentage of estimators that considered each month as a possible outcome of the project they estimated. An initial observation is that agreement among estimators is rather low. In the “2-months” condition, agreement only reaches 56%, at the 4 months line. For the control condition the maximum agreement is 50%, at the 4, 5 and 11 months lines. Finally, the maximum agreement for the “20-months” condition is 63%, at the 16 and 17 months lines.

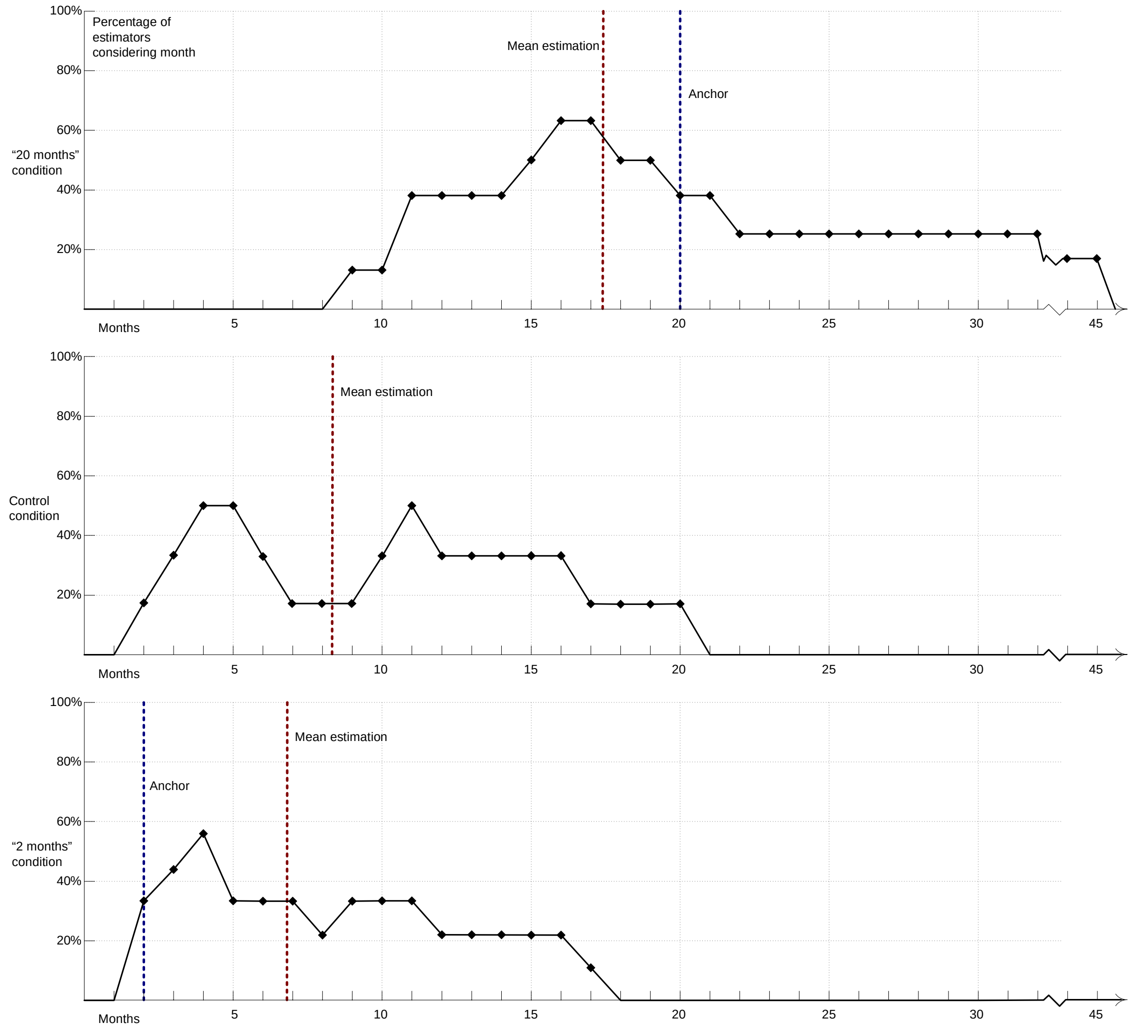

Figure 7.5 – Estimate ranges results, concentrated by condition

However, we should remember that participants in all conditions actually estimated the exact same project, and it would be interesting to merge the three charts and see the general agreement among estimators. Figure 7.6 shows that information.

Once that all estimators are considered, the maximum agreement is quite low (39%), and appears at two time points (at the 11 and 16 month lines). These results indicate that, were this project truly developed, *at least* 61% of the estimators would have been wrong in their predictions, a figure much lower than desired –but perhaps not perplexing considering how often estimates miss their targets in real software projects.

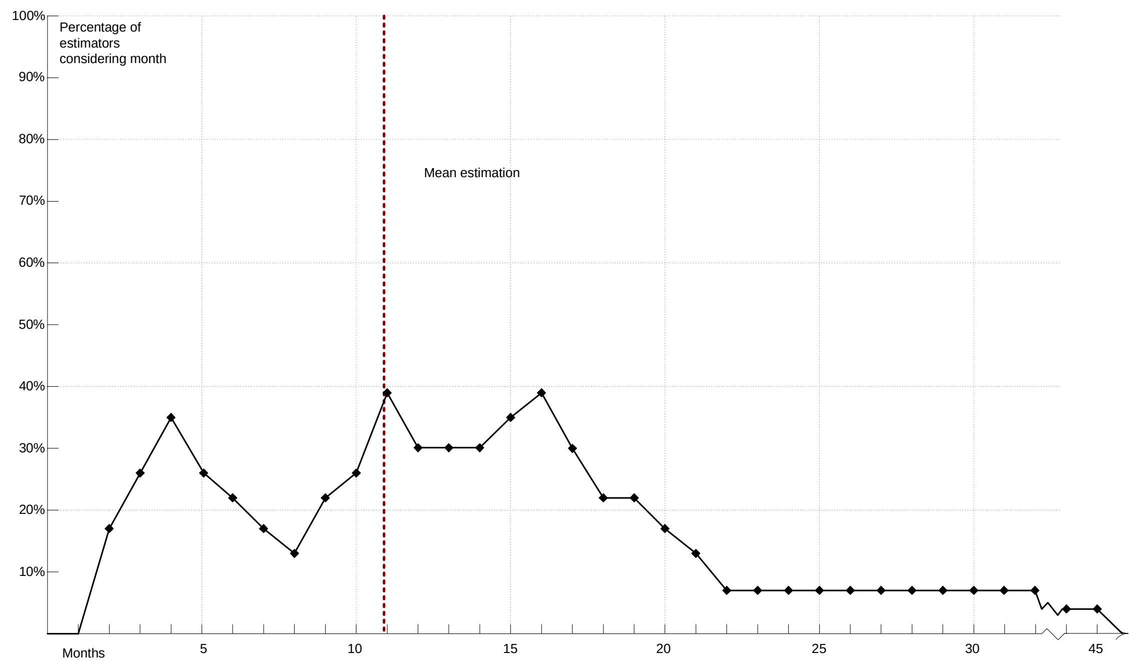

Figure 7.6 – Estimate ranges results, concentrated

Chapter 8

Discussion and Conclusions

This study demonstrated that anchoring and adjustment heuristics do take place in software estimation processes. When estimators are given a high anchor they are expected to produce an estimate significantly different than when they are given a low anchor or no anchor at all. The effect is maintained across experienced estimators and users of expert-based estimation techniques. Not enough data were collected to determine if the same effect occurs between low anchors and no anchors at all.

If we examine the nature of the observed anchoring and adjustment effect in this study, it is unlikely that estimates were biased toward anchors due to social reasons: participants did not need to comply with the anchor for either personal advancement or politics; the biasing comment was read instead of heard, and was labelled as a guess that should have no impact on professional estimators considerations. It seems therefore likely that the effect is indeed of a cognitive nature: Facing a complex problem, we tend to grasp a tentative solution and adjust it based on the facts we see.

The effects of anchoring are too strong to be ignored. On average, estimates on the high anchor condition were more than twice as long as those in the low anchor condition. The effects were so large than even the worst-case scenario produced by estimators in the low anchor condition is significantly more optimistic than the best-case scenario from the high anchor condition, which implies that estimators do not compensate the effect of anchoring and adjustment when given the opportunity to widen their estimates with confidence ranges.

Even though these effects were found to be statistically significant in all but the model-based techniques data points, the strongest effects were observed for expert-based techniques users.

The fact that no statistical difference was found between estimates in the low anchor condition and the control condition lead to several possible explanations for this phenomenon. There are at least three possible explanations:

- Estimators may be optimistic by nature, or they may intend to please by default. Therefore, in the control condition, participants may have felt the need to give optimistic, low estimates, substituting external anchoring effects with internal bravery.
- The value for the low anchor, 2 months, may have been chosen incorrectly. If the low anchor is too high, estimates in the control condition may be indistinguishable from those in the low anchor condition.
- An increased number of participants may be needed for the two data sets to separate more noticeably from each other.

This study was limited in the sense that its number of participants was relatively small, that it was concerned with only one software project, and that some estimators had only academic experience in software estimation. Its findings are therefore limited as well. However, its positive results produce further research questions that could be addressed with more experiments. Some of these future directions are:

- **Anchors with other estimation units:** Estimators were asked to provide their results in months. The possibility that this choice of units may have biased the estimates exists. Some estimators may think that “two months” sounds like a small quantity, but “nine weeks”, an approximate equivalent, seems longer. Therefore, it would be interesting to run this experiment again, but asking for the estimate in weeks and changing the conditions to "2 weeks" and "20 weeks", instead of months.

- **Estimates at different stages of the project lifecycle:** The requirements that participants used to reach their estimates were not detailed, and would correspond to documents produced at early stages of a project. We could address the question of whether anchoring and adjustment biases occur as strongly with a completely detailed specification and greater awareness of the environment in which the system would be developed.
- **Estimates as a factor in the final effort of software projects:** It has been noted that an estimate may influence the real effort of a project; shorter estimates causing shorter projects than those of longer estimates, due to resource constraints and a reduction of expectations [AM86]. A possible –but elaborate- experiment would attempt to measure the influence that an anchoring comment to an estimator early in a software project would have in the final product, in its number of functions and its quality.

The effects of anchoring and adjustment were observed to happen in this experiment and it is important to explore ideas to avoid or minimize their impact in software estimation tasks. The following suggestions may serve to reduce the problems caused by anchoring biases:

- **Shield estimators from anchors:** If possible, estimators should be protected from any anchor before they produce their estimates. However, this is not always feasible. Estimators tend to work in close contact to analysts, developers and clients, in environments where anchors are frequently formulated without regard for their impact.
- **Let estimators know that anchors may bias their own results:** Since anchoring and adjustment effects exist, it may be reasonable to warn estimators of their influence and hope they will consciously compensate for their effects. Although this suggestion seems sensible, previous studies have indicated that anchoring effects take place even when participants are warned from them [WHB93], so the suggestion may not be very helpful.

- **Give estimates with wide minimum-maximum ranges:** There seems to be too much optimism in early estimates. Reasonable studies [Boe81] indicate that confidence ranges of about 50% or 60% are adequate at early project stages, and estimators should resist the temptation to narrow their estimates. A problem with this suggestion is that managers have been observed to have a better opinion of estimators that give narrow but wrong estimates than of those that give wide but correct estimates, since they seem to be more experienced and knowledgeable.
- **Choose a development lifecycle where estimates are less relevant or incorrect estimates are a risk that is acknowledged:** Projects that follow a waterfall lifecycle are more vulnerable to incorrect estimates than projects that follow an iterative approach or a spiral model lifecycle. In general, to avoid the problems that incorrect estimations bring, of which anchoring and adjustment is only a part, less risky lifecycles should be chosen.

It is perhaps ironic that the software estimation exercise presented in this thesis does not have a “right” answer. Even if the project was developed, project goals may be partially set by their estimates, and a low estimate may produce a smaller product (or one with less quality) than a high estimate, as was mentioned above. If this is the case, the power of a seemingly innocuous anchor can shape a project as forcefully as its specifications.

Anchoring and adjustment biases may not be the biggest problems of software estimation. Considering that estimation is frequently done irrationally, that estimating processes tend to seem like bargaining matches, and that accuracy expectations of initial estimates are unreasonable, there are far too many more factors involved in flawed estimations that a misleading anchor. But the need to consider the effect of anchoring on estimates is nonetheless important if we intend to treat software estimation as anything more than guesswork.

References

[Alb79] Albrecht, A., “Measuring application development productivity”, *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, pp. 83-92, 1979.

[AM86] Abdel-Hamid, T., and S. Madnick, “Impact of schedule estimation on software project behavior”, *IEEE Software*, 3:4, pp. 70-75, 1986.

[Arm02] Armour, P., “Ten unmyths of project estimation”, *Communications of the ACM*, 45:11, 2002.

[AWA02] Aronson, E., T.D. Wilson and R.M. Akert, *Social Psychology*, Prentice Hall, 4th Edition, 2002.

[BAC00] Boehm, B., C. Abts and S. Chulani, “Software development cost estimation approaches – A survey”, *Annals of Software Engineering*, 10, pp. 177-205, 2000.

[Bai89] Baird, B., *Managerial Decisions Under Uncertainty*, Wiley, 1989.

[BCH+95] Boehm, B., B. Clark, E. Horowitz, C. Westland, R. Madachy and R. Selby, “Cost models for future software life cycle processes: COCOMO 2.0”, *Annals of Software Engineering, Special Volume on Software Process and Product Measurement*, 1995.

[BDR+03] Benediktsson, O., D. Dalcher, K. Reed and M. Woodman, “COCOMO-based effort estimation for iterative and incremental software development”, *Software Quality Journal*, 11, pp. 265-281, 2003.

[BH90] Blattberg, R.C., and S.J. Hoch, “Database models and managerial intuition: 50% model + 50% manager”, *Management Science*, 36, pp. 887-899, 1990.

[Boe81] Boehm, B., *Software engineering economics*, Prentice Hall, 1981.

[Bro95] Brooks, F., *The mythical man-month*, Addison-Wesley, 1995.

[BS93] Brown, N.R. and R.S. Siegler, “The role of availability in the estimation of national populations”, *Memory and Cognition*, 20, pp. 406-412, 1993.

[CB96] Chapman, G.B., and B.H. Bornstein, “The more you ask for, the more you get: Anchoring in personal injury verdicts”, *Applied Cognitive Psychology*, 10:6, pp. 519-540, 1996.

[DeM82] DeMarco, T., *Controlling software projects*, Prentice Hall, 1982.

[DL99] DeMarco, T. and T. Lister, *Peopleware*, Dorset House, 2nd Ed., 1999.

[Dol01] Dolado, J.J., “On the problem of the software cost function”, *Information and Software Technology*, 43, pp. 61-72, 2001.

[Fis82] Fischhoff, B., "For those condemned to study the past: Heuristics and biases in hindsight", In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.)*, Cambridge University Press, 1982.

[FWD97] Finnie, G.R., G.E. Wittig and J-M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models", *The Journal of Systems and Software*, 39, pp. 281-289, 1997.

[GJM04] Grimstad, S., M. Jørgensen and K. Mølokken-Østvold "Software effort estimation terminology: The tower of Babel", Submitted to *Information and Software Technology*, 2004.

[GM96] Gray, A. and S. MacDonell, "A comparison of techniques for developing predictive models of software metrics", *Information and Software Technology*, 39, 1996.

[Hel66] Helmer, O., *Social Technology*, Basic Books, 1966.

[HH91] Hihn, J., and H. Habib-agahi, "Cost estimation of software intensive projects: A survey of current practices", *International Conference on Software Engineering*, pp. 276-287, 1991.

[HK91] Heemstra, F.J., and R.J. Kusters, "Function point analysis: Evaluation of a software cost estimation model", *European Journal of Information Systems*, 1:4, pp. 223-237, 1991.

[Hog80] Hogarth, R., *Judgement and Choice*, John Wiley and Sons, 1980.

[HTA00] Hill, J., L.C. Thomas and D.E. Allen, “Experts’ estimates of task durations in software development projects”, *International Journal of Project Management*, 18:1, pp. 13-21, 2000.

[Hug96] Hughes, R.T., “Expert judgement as an estimating method”, *Information and Software Technology*, 38, pp. 67-75, 1996.

[Hum89] Humphrey, W., *Managing the software process*, Addison-Wesley, 1989.

[IEEE98] IEEE, *IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)*, 1998.

[Jon96] Jones, C., *Applied software measurement*, McGraw Hill, 1996.

[Jør04] Jørgensen, M., “A review of studies on expert estimation of software development effort”, *The Journal of Systems and Software*, 70, pp. 37-60, 2004.

[Jør03] Jørgensen, M., “How much does a vacation cost? or What is a software cost estimate?”, *ACM Software Engineering Notes*, 28:6, 2003.

[Jør97] Jørgensen, M., “An empirical evaluation of the MkII FPA estimation model”, *Norwegian Informatics Conference*, pp. 7-18, 1997.

[JRW00] Jeffery, R., M. Rune and I. Wieczorek, “A comparative study of two software development cost modeling techniques using multi-organizational and company-specific data”, *Information and Software Technology*, 42, pp. 1009-1016, 2000.

[JS04] Jørgensen, M., and D. Sjøberg “Expert estimation of software development work”, In *Software Evolution and Feedback*, Wiley, 2004.

[JS04b] Jørgensen, M., and D. Sjøberg “The impact of customer expectation on software development effort estimates”, *International Journal of Project Management*, 22, pp. 317-325, 2004.

[JS01] Jørgensen, M., and D. Sjøberg “Impact of effort estimates on software project work”, *Information and Software Technology*, 43, pp. 939-948, 2001.

[JTM04] Jørgensen, M., K.J. Teigen and K. Mølokken-Østvold “Better sure than safe? Over-confidence in judgement based software development effort prediction intervals”, *The Journal of Systems and Software*, 70, pp. 79-93, 2004.

[Kem87] Kemerer, C.F., “An empirical validation of software cost estimation models”, *Communications of the ACM*, 30:5, 1987.

[KPM+02] Kitchenham B., S.L. Pfleeger, B. McColl and S. Eagan, “A case study of maintenance estimation accuracy”, *Journal of Systems and Software*, 64:1, pp. 57-77, 2002.

[KST82] Kahneman, D., P. Slovic and A. Tversky (Eds.), *Judgment under uncertainty: Heuristics and biases*, Cambridge University Press, 1982.

[KT82] Kahneman, D., and A. Tversky, “On the psychology of prediction”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.)*, Cambridge University Press, 1982.

[Mad94] Madachy, R., “A software project dynamics model for process cost, schedule and risk assessment”, *Ph.D. Thesis, University of Southern California*, 1994.

[MJ04] Mølokken-Østvold, K., and M. Jørgensen, “Group processes in software effort estimation”, *Empirical Software Engineering*, 9, pp. 315-334, 2004.

[MS01] Mussweiler, T., and F. Strack, “The semantics of anchoring”, *Organizational Behavior and Human Decision Processes*, 86:2, pp. 234-255, 2001.

[NN87] Northcraft, G.B., and M.A. Neale, “Experts, amateurs and real estate: An anchoring and adjustment perspective on property pricing”, *Organizational Behavior and Human Decision Processes*, 39:1, 1987.

[Osk82] Oskamp, S., “Overconfidence in case-study judgments”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.)*, Cambridge University Press, 1982.

[Pay96] Paynter, J., “Project estimation using screenflow engineering”, *International Conference on Software Engineering: Education and Practice*, 1996.

[Pen95] Pengelly, A., “Performance of effort estimating techniques in current development environments”, *Software Engineering Journal*, 10:5, pp. 162-170, 1995.

[PM92] Putnam, L. and W. Myers, *Measures for Excellence*, Yourdon Press Computing Series, 1992.

[RA82] Ross, L., and C.A. Anderson, “Shortcomings in the attribution process: On the origins and maintenance of erroneous social assessments”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.),* Cambridge University Press, 1982.

[Rub83] Rubin, H., “ESTIMACS”, *IEEE Software*, 1983.

[SB95] Subramanian, G.H. and S. Breslawski, “An empirical analysis of software effort estimate alterations”, *Journal of Systems and Software*, 31, pp. 135-141, 1995.

[SFL82] Slovic, P., B. Fischhoff and S. Lichtenstein, “Facts versus fears: Understanding perceived fears”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.),* Cambridge University Press, 1982.

[Sin82] Singer, M., “The vitality of mythical numbers”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.),* Cambridge University Press, 1982.

[SS97] Shepperd, M. and M. Schofield, “Estimating software project effort using analogies”, *IEEE Transactions on Software Engineering*, 23:12, 1997.

[Sta94] Standish Group, The, *Charting the seas of information technology*, The Standish Group, 1994.

[TK82] Tversky, A., and D. Kahneman, “Availability: A heuristic for judging frequency and probability”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.),* Cambridge University Press, 1982.

[TK82b] Tversky, A., and D. Kahneman, “Belief in the law of small numbers”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.),* Cambridge University Press, 1982.

[TK82c] Tversky, A., and D. Kahneman, “Judgment under uncertainty: Heuristics and biases”, In *Judgment under uncertainty: Heuristics and biases; Kahneman, D., P. Slovic and A. Tversky (Eds.)*, Cambridge University Press, 1982.

[WJ99] Walkerden, F., and R. Jeffery, “An empirical study of analogy-based software effort estimation”, *Empirical Software Engineering*, 4, pp. 135-158, 1999.

[WHB93] Wilson, T.D., C.E. Houston and N. Brekke, “A new look at anchoring effects: Basic anchoring and its antecedents”, *Journal of Experimental Psychology: General*, 125:4, pp. 384-402, 1993.

[WRH+00] Wohlin, C., P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 2000.

Appendix 1

Participant Results

Part. #	Condition	Estimate (mts.)	Range (+/-)	Experience	Method
1	20 months	16	20%	Academic	Expert
2	2 months	10	10%	Exper., med/lrg	Expert
3	20 months	18	15%	Exper., med/lrg	Expert
4	2 months	4	50%	Academic	Expert
5	Control	9	20%	Exper., med/lrg	Expert
6	2 months	13	20%	Exper., small	Model
7	20 months	10	15%	Academic	Model
8	2 months	6	20%	Exper., small	Expert
9	20 months	28	60%	Exper., small	Model
10	Control	15	30%	Academic	Model
11	20 months	12	10%	Exper., med/lrg	Expert
12	2 months	4	10%	Exper., small	Expert
13	20 months	15	10%	Exper., med/lrg	Expert
14	2 months	12	40%	Academic	Model
15	Control	4	20%	Academic	Expert
16	Control	4	50%	Academic	Model
17	2 months	3	20%	Academic	Expert
18	20 months	16	20%	Exper., small	Expert
19	Control	5	20%	Exper., med/lrg	Expert
20	Control	13	25%	Exper., small	Expert
21	2 months	3	25%	Academic	Expert
22	20 months	24	35%	Academic	Model
23	2 months	6	50%	Exper., small	Expert

Appendix 2

Experiment Instructions

Project Estimation Experiment

Participant #: ______

INSTRUCTIONS

Along with this sheet you will find two documents concerning a possible new software development project. The first of them, "Software Requirements Initial Report", is a preliminary requirements document that explains the functionality expected of the system, without getting in too much detail. The second document, "Project Setting", will give you information about the client organization and the team in charge of the project, which you may found relevant for the exercise as well.

Your task is to read thoroughly both documents, perform any calculations you wish, use estimation tools if desired or needed, and respond to the two questions below. After you answer the questions you will be given a questionnaire about the task you performed and your previous experience in software engineering.

QUESTIONS

1.- Give your estimate for the duration of the project described in the attached documentation, in months, to the nearest integer:

I ESTIMATE THE PROJECT WILL TAKE _______ MONTHS TO DELIVER.

2.- Justify your estimation. Try to justify it in such a way that a reader may understand and follow your reasoning and probably reach the same conclusion. If you need it you can use the back of this paper and as many additional sheets as you want to make your point.

Appendix 3

Requirements Specification

Toronto Foreign Trade Statistics System

Software Requirements Initial Report

1. Introduction

This document describes the requirements of a system to help the Toronto Foreign Trade Agency, a municipal public office, to analyze and report statistics on foreign trade of Toronto- and GTA-based companies.

1.1 The Project at a Glance

The software to be produced will be named *Foreign Trade Statistics System*, and will be referred to as FTSS in the rest of this document.

The main purpose of the FTSS is to receive, store and process federal data about foreign trade engaged by Toronto- and GTA-based organizations; serving as a statistics generation tool for users to analyze its information and produce official, municipality level foreign trade reports.

The client organization currently owns and operates a software tool that is used for this purpose; but its age and its design inadequacies became evident as the database size and query complexity increased, and it is now obvious that it will be obsolete relatively soon. The FTSS will therefore phase out the previous system.

The high-level goals of the new system are:

a. To reduce the time it takes for office staff to produce and deliver *standard* reports; from an average of 1 hour to an average of 10 minutes. The following subsection has more details on standard reports.
b. To reduce the time it takes for office staff to produce and deliver *custom* reports; from an average of 12 hours to an average of 15 minutes. The following subsection has more details on custom reports.
c. To increase the confidence of the reports; from the present 97.5% (one faulty report in forty) to 99.9% (one faulty report in one thousand) or better.
d. To provide a new web service that allows Internet users to post –relatively- simple queries to the system through the Toronto Foreign Trade Agency website.
e. To reduce system downtime, through proper software design and a robust implementation, to at most an hour of downtime each three months.
f. To allow the Statistics Staff, through the fulfillment of the previous objectives, to spend more of their time on statistics analysis and less on statistics production.

Note that the system currently being used has the functionality to produce standard reports. Although the system is slow, the reasons why it presently takes an average of one hour to produce them are the need to review them thoroughly because of a history of error-prone reports, and the need to format them in the way upper management wants them (with neatly arranged and coloured Excel files). Note as well that the time it takes to deliver custom reports is so long (12 hours on average) because the present system does not provide the functionality to produce them at all. The statistics staff have to plunge into reams of reports and perform calculations by themselves to obtain the desired information, and then format it the way the Agency requires.

1.2 Definitions, acronyms, and abbreviations

The following alphabetically sorted definitions may help to better understand this document and the specifications:

- *Custom report* – A statistical report that is not part of the basic set of *standard* reports (see below) and which links any two or more relevant factors of a statistic. Non-comprehensive examples of custom reports are:
    - A list of the top ten countries to which there were exports from Scarborough during 2003, with export volumes.
    - Names of the top five companies from the GTA that imported European machinery in the last five years.
    - Transaction amount of exports from the GTA grouped by company size categories and presented a month per column, from January 2002 to date.
    - A comparison of the volume of imports processed at each official customs port, with yearly columns from 1999 to 2003.
    - A matrix showing the amount for the transactions made from every region of the world (Europe, Latin America, etc. –defined by user) to each section of the Greater Toronto Area for 2002.
    - A list of the 100 most exported and 100 most imported goods (from an official federal catalogue), based on number of transactions, during Jan-Apr 2004.
- *Exports* – Commercial transaction in which goods are transferred from Canada to another country.
- *Foreign Trade* – An interchange of goods and money between two companies based on different countries. We will be only concerned with the foreign trade between GTA-based companies and foreign companies.
- *FTSS* – Foreign Trade Statistics System; the software product this document is concerned with.
- *Imports* – Commercial transaction in which goods are transferred to Canada from another country.
- *Standard report* – A statistical report that is typically produced each month with the new information available. It is delivered as a neatly formatted MS Excel file, and it is sent to the media and compiled in regional statistics books. There are 32 standard monthly reports, and 18 more standard annual reports. Examples of standard reports are:
    - A list of total exports grouped by country, from the beginning of this year to date, compared to the same period of last year.
    - A list of total imports grouped by type of goods (predefined, standard) for each month of the present year.
- *Toronto Foreign Trade Agency* – The client organization.

1.3 Overview

The rest of this document contains the required specification for the software. Section 2 presents the requirements of the system, and Section 3 mentions other necessary considerations.

2. Requirements description

The description of the intended product is detailed in the following subsections.

2.1 FTSS and its environment

The diagram in Figure 1 explains the intended relationship of the FTSS and its environment.

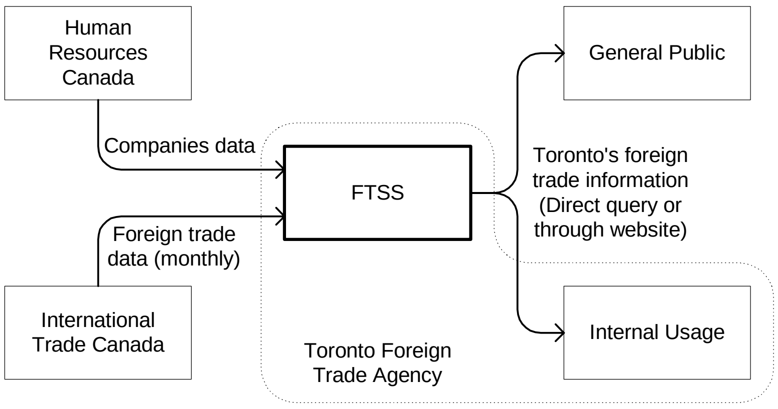
Human Resources Canada
Companies data
Foreign trade data (monthly)
International Trade Canada
FTSS
Toronto Foreign Trade Agency
General Public
Toronto's foreign trade information (Direct query or through website)
Internal Usage

Figure 1 – FTSS Product Perspective

The main inputs to the FTSS come from two federal agencies. One of them, Human Resources Canada, provides company data regarding every organization registered in Toronto and the GTA. The other, International Trade Canada, provides monthly foreign trade data of Toronto and GTA-based companies. It is the job of the system and its users to analyze and process the data, generate reports based on them, and deliver the reports to the general public or for internal usage.

The following items are relevant to this perspective of the system:

a. Both the companies data and the foreign trade data are received by courier, on a CD with predefined format files.
b. The companies data is received whenever it is requested; typically once every two months.
c. The foreign trade data is received monthly.
d. The Trade Agency generates standard official reports each month (see section 1.3). These reports are used by other staff at the Agency, and are available through the Agency website and in print.
e. The Agency generates custom reports as well (see section 1.3), both for the general public and (more commonly) for internal usage.
f. The Statistics Office, which produces these reports, is only one of several departments of the Agency. Other foreign trade related departments within the Agency include: GTA Promotion, Foreign Trade Training and Federal Link offices. These departments need the information generated by the Statistics Office, and often request custom reports from it.

2.2 Product functions

The main functional requirements of the system are:

a. The FTSS shall provide an interface to receive Companies Data from a CD, detect and filter out inconsistencies in the CD file (which are known to happen) and update its own companies information. Company data include names, addresses, telephone and fax numbers, e-mails, company size and main businesses. This interface shall be as easy to use as executing a menu option.

b. The FTSS shall provide an interface to receive Foreign Trade Data from a CD, which shall work in the same way as the Companies Data interface. Foreign Trade Data consist of one record per transaction, and each transaction has the fields: Date of transaction, type of transaction (import/export), GTA company in charge of the transaction, type of goods traded (from a catalogue), units and quantity, transportation method, country of origin/destination, customs office that dealt with the merchandise and monetary value of the transaction. Sometimes International Trade Canada sends transaction information that does not correspond to GTA companies. These records should be filtered out of the database.

c. The FTSS shall provide the capability of obtaining standard reports in 2 minutes in average (without including user revision of the report, which will take about 8 more minutes approximately). Standard reports shall be formatted in a MS Excel file format by the system. Please see Section 1.3 for more details. This function should work the following way: The user will be presented a choice of all possible standard reports (50 options in total), will choose one of them, and select the time period for which (s)he wants the information. The software should obtain the necessary information and prepare a MS Excel file with all proper format, ready to be revised and submitted.

d. The FTSS shall provide the capability of obtaining custom reports in 5 minutes in average (without including user revision of the information, which will take about 10 more minutes, approximately). Custom reports are presented in MS Excel files, but they do not need the fancy formatting that standard reports feature. Please see Section 1.3 for more details. This function should work the following way: The user chooses a type of report (list, cross table, etc), chooses the grouping of information (s)he is interested in, the time periods which should be included and any relevant constraints. Choices for information grouping are countries, world areas, GTA sections, types of goods, company size, customs offices, method of transportation, company name and postal code. Examples of constraints are: [Types of goods: Food]; [Company size: Not small]; [Method of transportation: Train or ship]; and [Postal code: {list of codes}]. Furthermore, the user should be given the option to save the settings of the custom report (s)he created, so that (s)he can go back to it later, regenerate or reprint it, or change some of the constraints to produce a different report.

e. The FTSS shall provide a web service that allows Internet users to post queries, similar to those prepared by standard reports, and obtain official information on foreign trade.

f. The FTSS shall provide database backup and restoration features.

g. The FTSS shall provide means to update the basic information it needs to work. This information shall be available to users to add new elements, update existing elements or delete no longer needed elements. The types of elements to be considered are:
   - ❑ Countries
   - ❑ World areas,
   - ❑ GTA sections
   - ❑ General types of goods
   - ❑ Detailed catalogue of types of goods (with more than 50,000 entries)
   - ❑ Valid trading units (kilograms, pounds, etc.)
   - ❑ Recognized customs offices
   - ❑ Means of transportation.

h. The FTSS shall provide detailed company data. In some situations, the Agency finds that a company handles part of its foreign trade transactions from the GTA and part from other regions. If this is the case, the user should be able to update the company information, and to specify to the system that only (a) a percentage of the total transactions of the company, or (b) a list of detailed types of goods, should be considered when producing foreign trade information from that company.

i. The FTSS shall provide user access control and permissions management, since the information it handles is confidential and delicate. As to this moment it has not been defined if the data will have to be encrypted to protect it.

3. Other considerations

This section provides necessary, non-functional information on what is expected from the system and on characteristics of its environment

3.1 Reliability of reports

Reliability of reports has a high priority. The system should guarantee total accuracy in at least 99.9% of the reports it generates. Consider that the information that the Agency receives is sporadically inconsistent. If that is the case, the system should detect the inconsistencies and notify the users about them.

3.2 Information volume

The system should be able to handle at least 15 years of information without failing the time constraints expressed in the objectives section of this document. It is important to consider that 15 years of information may easily extend to several tens of gigabytes of memory.

3.3 User interface

The client organization expressed repeatedly that the user interface of the system should also be a priority, and that it should not only be friendly and agile, but elegant and adherent to the color code of the Agency. The software should be usable by an average user with two two-hours training sessions up to 95% of its functionality. The system should also provide clear and detailed online help so that a user that has had only two training sessions can find his way on the system by himself.

3.4 User characteristics

The intended users of the FTSS are a subset of the staff of the Agency, and the general public through the Agency website. The expected characteristics of the users are:

- ❑ Familiar with the Windows operating system, which presently is the only one used in the Statistics Office
- ❑ College or university educational level
- ❑ Not familiar with any database system nor programming language

It is expected that about 3 persons will use the software concurrently. However, the software should be designed to support at least 10 simultaneous users.

3.5 Programming language constraints

The programming language to be used has been already selected to be Visual C++ .NET, as a contract condition pushed by the IT staff of the Agency. As a similar condition the backend for the database will be SQL Server 2000. The IT staff did not express an opinion on the choices for script languages for the web services required.

3.6 Process requirements

Once that the Agency and the development team agree on time and cost figures for the software, the project leader will need to present weekly progress reports to the statistics office manager, both in writing and in person.

Appendix 4

Project Setting

Project Setting

The following text is included because it may give you information that you would otherwise perceive about the client organization and the development team if you dealt with them, but that is not included in the Software Requirements document.

About the client organization

The Agency is a corporate-like office, with a tendency to have plenty of meetings and concern for hierarchical levels. Complex matters must be approved by upper management, and their decisions may be delayed if those who take them are unavailable due to travel –which happens frequently. On the other hand, office workers are slightly understaffed, constantly working under pressure and performing quickly so the office may respond to the constantly shifting international situation (and to the whims of upper management).

Some people on the Agency relevant for the project are:

Statistics Office staff (three persons)

- Will be main users of the system
- Are very concerned about the accuracy of the output.
- Will be the main source of information and business knowledge.
- Frequently work under pressure

- Personnel rotation is high

Quotes from preliminary interviews with the staff:

"Right now, with the system we have, I must check, double-check and check again the results of my calculations to avoid any errors"

"The system we're using is horribly slow, it crashes all the time and it doesn't filter out bad data we receive from the federal offices"

"If people from other departments come and ask for a complex statistic they get bothered if we tell them the answer will be ready in four or five hours – but right now that's all we can offer, even putting everything else we're doing aside"

"One error on an official report and I lose my job!"

Statistics Office manager

- Manages the statistics staff
- Concerned about saving face with upper management

Quotes from preliminary interviews with the manager:

"I admit I have no experience with software projects, but I guess this will take about 2 months to finish. I may be wrong of course, we'll wait for your calculations for a better estimate."

"Your product should have a great look, really slick and professional. This is very important. Of course, no errors is a top priority; but if the thing looks ugly, or if it doesn't stick with the Agency colour code, when the Chair sees it he won't like it, and you guys and me will have a hard time trying to explain it does work well. They look forward to impress the people from other municipal foreign trade offices with this system, you know."

Systems administrator

- Has answered to every question promptly, but not helpfully.
- Is specifically concerned about the part of the project that deals with the office's website.
- Along with the IT staff, has cornered the choice of programming language and database engine.

About the development team

The project will be developed by a predefined team. Your estimate should adjust to the fact that only these persons will be available to work on it:

Project leader

- He’s an experienced developer that will perform as project leader for the second time (the first time his project was moderately successful – delivering a good amount of required functionality, on time but slightly over budget).
- He will be the face of the development company with the client.
- He could perform some duties as senior developer.
- Has good experience with the required programming language and database system.

Senior developer

- He got his experience developing scientific software for several years, and afterwards shifted to custom-made applications and has participated in a couple of developments of the kind.
- Has thorough experience with language and database of choice

Junior developer 1

- She was brought to the project for her interest and experience in database issues, which she seems to handle well.
- She has some experience with the language of choice, though she is not an expert.

Junior developer 2

- He has some experience on web applications development.
- His interests are human-computer interaction and web applications.
- Has a very introductory-level experience with the language of choice, and will need to have training during the first weeks of the project.
- He will only collaborate on this project half-time, (he is also required in a second project).

Quality assurance / tester

- She has been lead tester of three projects before this one.
- Has had minor negative personal issues with the senior developer in the past, apparently sorted out.
- Her tests have previously focused on software robustness and heavy database use.

- She will only collaborate on this project half-time, as she will fulfill the QA role in another project at the same time.

No one in the team has previously worked in international trade environments; knowledge of the field will have to be obtained through the client organization.

Appendix 5

Questionnaire

Participant #: ______

QUESTIONNAIRE

Please respond to the following questions as accurately as possible:

1.- I think the estimation I performed was… (circle a number)

Very unreliable ->	1	2	3	4	5	6	7	<- Very reliable

2.- I think that if this project was really developed, my estimate might be off by as much as _____ %

3.- My previous estimation experience includes (check all that apply)

___ Involvement in estimation of medium or large software projects

___ Involvement in estimation of small software projects

___ Courses that included software estimation as a course topic

___ I witnessed software estimation processes, although I was not involved

___ Self-learning

___ Other, specify: ____

-or-

___ I have no previous experience on software project estimation

4.- I felt the documentation was… (circle)

Very uninformative ->	1	2	3	4	5	6	7	<- Very informative

If you answered 3 or less, please specify:

5.- Explain your strategy to estimate software development projects