Dissertation

Analogy-Based Software Cost Estimation: Improving Explicability, Effectiveness and Efficiency

ausgeführt zum Zweck der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao. Univ. Prof. Dr. Stefan Biffl
Institut 188
Institut für Softwaretechnik und Interaktive Systeme

eingereicht an der Technischen Universität Wien,
Fakultät für Informatik

von
Dipl.-Ing. Mag. Martin Auer
Mat.-Nr.: 9326774
Schloesselgasse 19/13
A-1080 Wien

Wien, am 30.5.2004 [illegible] ....

We had the sky up there, all speckled with stars, and we used to lay on our backs and look up at them, and discuss about whether they was made or only just happened.

—Mark Twain, "Adventures of Huckleberry Finn"

Working on this thesis' topics together with a young research team at the Institute of Software Technology has been a great pleasure. I'd especially like to thank Stefan Biffl for giving advice and feedback and for playing devil's advocate in the initial stages of ideas.

Several graduate students worked with me on the same topics and ensured that we'd meet our submission deadlines—my thanks go to Bernhard Graser, Christoph Becker, and Oliver Hödl.

Andreas Rauber helped us kindly with his digital library tool; Matthias Heindl provided us with invaluable requirement documents; Dietmar Winkler often relieved me from other duties—thank you!

Martin Auer, Vienna, 2004

Contents

List of Tables

List of Figures

Abstract

Software cost estimation is a fundamental task in software project portfolio management. It is vital to many prominent portfolio decisions, like resource allocation, bidding, start scheduling, or risk management. However, the estimation process is complex due to the involvement of different stakeholders with varying priorities, difficult-to-re-enact estimate creation, the high influence of human experience, or software measurement problems.

This thesis proposes methods to significantly improve three main cost estimation quality criteria with a particular estimation method, the analogy-based estimation approach. This approach tries to create estimates based on historic experiences of "similar" projects, whereby similarity is usually defined as Euclidean distance based on several project attributes.

The improvement criteria are: estimation effectiveness, i.e., accuracy and reliability; estimation explicability, i.e., transparency and acceptability; and estimation efficiency. Accuracy and reliability describe how close estimates are to actual costs after project completion, and how likely estimates will differ from actual costs. Transparency and acceptability determine how straightforward it is to understand a model and to accepts its estimate creation procedure, and, ultimately, its estimates. Finally, efficiency describes the costs that arise during the estimation process, for example, during the collection of software metrics to calibrate estimation models.

To improve estimation quality, this thesis addresses three stages of the overall estimation process specifically: (1) project measurement data collection (collect), (2) estimate proposal creation (create), and (3) the sanity-check of the estimate proposals (check).

Poor estimation accuracy leads to sub-optimal resource allocation and to greater risks in software development. To achieve higher estimation accuracy, this thesis focuses on stage 2 of the estimation process, estimate creation. A prominent type of cost estimation—the analogy-based estimation method—is enhanced by taking into account the varying influence of historical project measures or features on the estimate. An extensive search for optimal weights of these features yields higher estimation accuracy and reliability.

Intransparent estimation methods do not unleash the potential of estimators' experience, and inhibit the application and acceptance of formal methods. To achieve higher estimation transparency, both stage 2 and 3 of the estimation process, estimate creation and sanity-check, are addressed. In stage 2, the use of more appropriate feature weights causes the estimation model to be less volatile, the estimate creation to be better repeatable and thus more plausible. In stage 3, a graphical representation of historical project analogies is used to allow for more intuitive understanding of an estimate's plausibility; it is based on multi-dimensional scaling methods.

Finally, inefficient estimation procedures are expensive, and either unlikely to be implemented, or difficult to sustain over a long period of time. To achieve higher estimation efficiency, in stage 1 of the estimation process, an alternative way of collecting project measurement data is proposed and analyzed. A project portfolio's use cases descriptions are used to implicitly assess use case similarity based on their textual representation; a variant of self-organizing maps is used for this purpose.

The thesis applies all proposed approaches to real-world data sets. It quantifies the increase in accuracy and reliability, as well as the reduction in the model's volatility, and the approximation quality of the graphical representation of the analogies.

Main results are:

- Significantly increased estimation accuracy and reliability: by taking into account the varying importance of different project features, more precise estimates are achieved.

- Indication of a barrier for analogy-based estimate proposal quality: the extensive search for feature weighting yields optimal feature weights that can't be improved—not even by experts knowing about the features' influence.

- A significantly less volatile, and thus more transparent model: traditional approaches to feature weighting would yield highly volatile behavior of the weights over a typical portfolio lifetime, i.e., as projects are added subsequently to the portfolio; this thesis' approach yields more stable weight behavior, increasing the confidence in the estimation model.

- A high quality of the graphical representation of project analogies: real-world portfolio data can be visualized with a high approximation quality, giving insight into the analogies' relative importance.

- A cost-saving implicit way of determining analogies in historic project data: textual use case similarities yield meaningful requirement cluster, allowing for fast and intuitive browsing of large requirement repositories.

The thesis thus indicates the benefits of the application of powerful mathematical methods to concrete real-world aspects of cost estimation, calling for further integration of estimation aids to foster their application.

Zusammenfassung

Softwarekostenschätzung ist eine fundamentale Aufgabe im Management von Softwareprojektportfolios. Sie ist unabdingbar bei vielen wichtigen Portfolioentscheidungen, z.B. Ressourceneinteilung, Bieten, Projektstartplanung, oder Risikomanagement. Allerdings ist der Prozess der Schätzung komplex wegen der Einbeziehung verschiedenen Projektteilnehmer, wegen schwer nachvollziehbarer Erzeugung der Schätzungen, wegen des Einflusses der menschlichen Erfahrung, und wegen Softwaremessproblemen.

Diese Arbeit schlägt Methoden vor um drei Hauptqualitätskriterien von einer besonders gut einsetzbaren Schätzmethode, der analogie-basierten Schätzung, signifikant zu verbessern: Schätzgenauigkeit, Transparenz der Schätzung, und Schätzeffizienz. Das wird erreicht indem drei Stufen eines allgemeinen Schätzprozesses spezifisch addressiert werden: (1) Messdatensammlung, (2) Erzeugung des Schätzvorschlags, und (3) Evaluierung der Schätzung.

Um höhere Schätzgenauigkeit zu erzielen wird der Fokus auf Stufe 2 gelegt, Erzeugung des Schätzvorschlags. Die analogiebasierte Schätzmethode wird erweitert durch Einbeziehung des unterschiedlichen Einflusses von historischen Projekteigenschaften auf die Schätzung. Eine extensive Suche nach optimalen Gewichtungen erzielt höhere Genauigkeit und Zuverlässigkeit.

Um die Transparenz der Schätzung zu erhöhen, sowohl in Stufe 2 als auch in Stufe 3, werden Erzeugung und Evaluierung verbessert. In Stufe 2 veranlassen die optimalen Gewichtungen der Projekteigenschaften ein weit weniger volatiles Schätznodell beim inkrementellen Wachsen des Portfolios durch das Hinzufügen neuer Projekte; damit wird die Erzeugung der Schätzung nachvollziehbarer und plausibler. In Stufe 3 erleichtert eine grafische Darstellung historischer Projektanalogien ein intuitives Verständnis der Schätzqualität; diese basiert auf Methoden der multidimensionalen Skalierung.

Um eine höhere Effizienz in der Schätzung zu erzielen, werden in Stufe 1 alternative Wege zur expliziten Sammlung von Softwaremetriken aufgezeigt und analysiert. Die Anwendungsfallbeschreibungen eines Projektportfolios werden verwendet um implizit die Ähnlichkeit von Anforderungen festzustellen. Eine Variante von "self-organizing maps" wird hierzu verwendet.

Alle vorgestellten Verfahren werden an echten industriellen Datensätzen und Artefakten angewendet. Die Verbesserungen der Schätzgenauigkeit und -zuverlässigkeit, die Reduktion der Modellvolatilität, und die Annäherungsqualität der grafischen Darstellung werden quantifiziert.

Hauptergebnisse sind: signifikant erhöhte Schätzgenauigkeit und -zuverlässigkeit; eine Schranke für maximale Schätzqualität bei analogie-basierter Schätzung; ein weit weniger volatiles, und damit transparenteres Modell; eine hohe Annäherungsqualität der grafischen Darstellung der Projektanalogien; und ein kostensparender Weg um implizit Analogien in historischen Portfoliodaten zu ermitteln.

Chapter 1

Introduction

Software has become a main pillar of our society and economy and is likely to increase its influence in the decades to come. Ever larger parts of our lives are deeply affected by advances in communication and software technology; ever more people and companies work on the development of new software solutions.

Software development changed rapidly in the last 50 years: from singular development projects to product lines, from ad-hoc creation of programs to process-driven development environments, from expert-centric special solutions to mass market applications [Kru00, ZBGK01]. Thus, typical software development companies are long dealing with software project portfolios rather than single projects, continually facing portfolio decisions: project selection and start scheduling, resource planning and allocation, bidding, outsourcing, development to research ratio setting, etc. Efficiently managing these increasingly complex project portfolios is a key success factor.

A fundamental aspect of most portfolio decisions is the estimation of costs and efforts. It is fundamental to determine benefits of investments and risks of developments, it is key to efficient resource allocation.

How can decisions and the supporting task of estimation be linked? Consider the following generic decisions process model, also

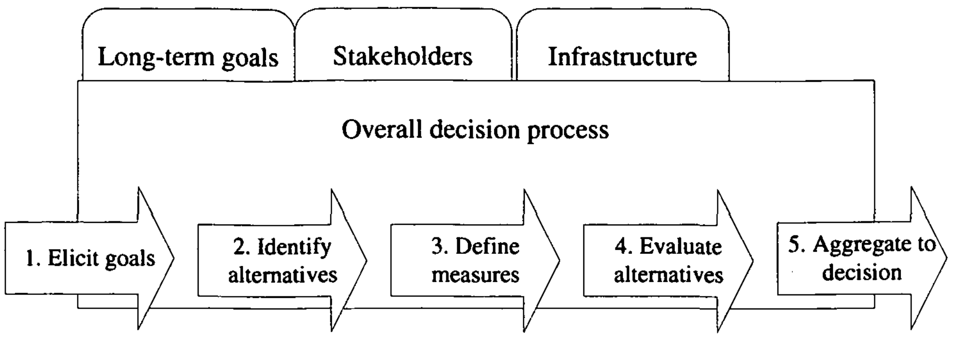

Figure 1.1: Main steps of decision process [Bif04]

depicted in figure 1.1:[1]

1. *Elicit goals and preferences.* In accordance to a company's vision and long-term objectives, specific business goals and policies are defined.

2. *Identify alternatives.* Various ways to achieve these goals, as well as a broad range of possible scenarios are outlined.

3. *Define performance measures.* To compare and evaluate the alternative options, measures for their performance, as well as for success or failure, are defined.

4. *Evaluate alternatives.* Using these performance measures, the decision stakeholders evaluate the possible alternatives in the context of their interests.

5. *Aggregate results to overall decision.* Finally, the stakeholders' evaluation must lead to an overall decision; this process may—but need not—rely on the stakeholders to agree.

[1]The generic decision model is derived from the decision framework CIDECS—"Collaborative Interactive DECision Support"—currently developed at the Institute of Software Technology, Vienna University of Technology [Bif04].

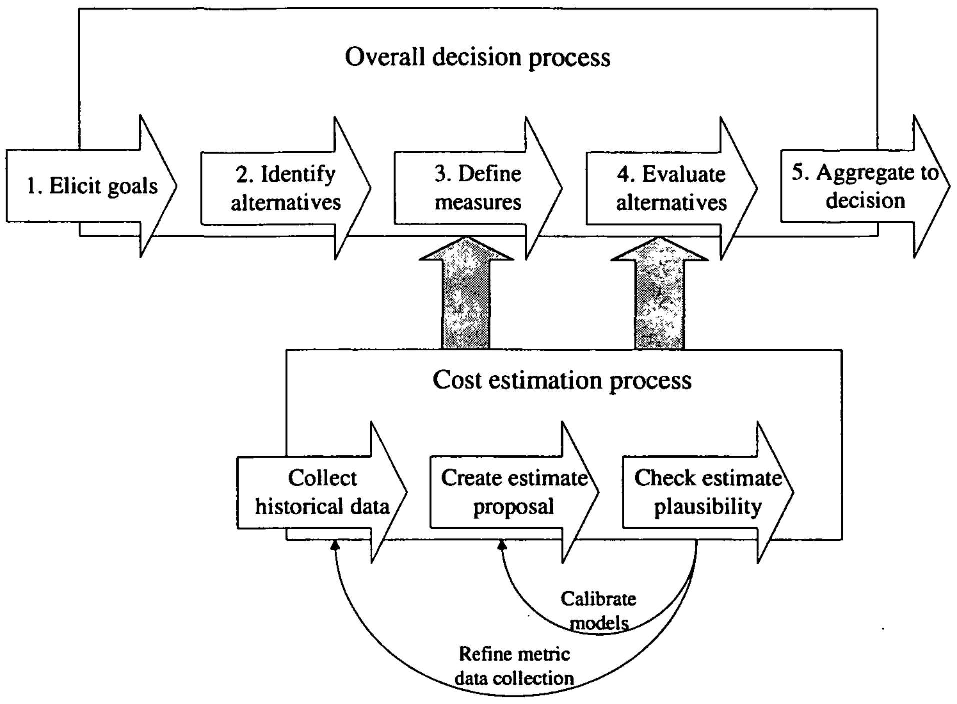

Figure 1.2: Cost estimation supports portfolio decision making

Estimation and a corresponding estimation method is particularly related to steps 3 and 4. The estimation model used influences the definition of performance measures by its perceived accuracy and reliability. The subsequent evaluation of alternatives, in most cases a collaborative process of different stakeholders, involves evaluating estimate proposals; critical criteria here are an estimation method's acceptance and transparency.

To put flesh on this decision model, and to illustrate the tight relation between important portfolio decisions and cost estimation, several decisions are given in the following.

- *Resource allocation, project selection and scheduling.* Choosing the projects that maximize a portfolio's total value is the most prominent portfolio decision. Extensive literature exists describing project selection and resource allocation methods for R&D projects in terms of decision rules and programming models; reviews on existing literature are, for example, provided in [Lie98, KI94, HS99, SH03].

Kira et al. [KKMG90] present a specific decision support system for R&D project selection under uncertainty based on integer linear programming and Monte Carlo simulation techniques. Kocoaglu and Iyigun [KI94] categorize earlier research into decision event models and decision process models and develop a decision support system to assist management in strategic project selection and resource allocation; they follow a decision process model employing multi-objective integer programming methods. Linton et al. [LWK+00, LWM02] propose a method for the analysis, ranking and selection of R&D projects. Other methods include multiple attribute game theory [AKK93], simulation [RGC00, VL88], and heuristics [VV95].

Lukesch [Luk00] demonstrates the practical application of employing standard risk-benefit diagrams to identify the most valuable projects in a portfolio. Risk-benefit and cost-benefit diagrams and variations thereof, like risk-reward bubble diagrams, are a common and widely applied visualization technique often used in portfolio management. Projects are ordered along the two dimensions benefits and costs versus risks. The portfolio can thus be partitioned into four main classes, ranging from projects with low costs and negligible benefits to costly projects with high benefits. The median provides an additional intuitive way to divide the portfolio in two classes, namely profitable and non-profitable projects. Projects above the median should be realized for their potential benefits, those clearly beneath should be significantly changed or abandoned, whereas projects near the median might be worth further consideration. Imperative for a proper assessment of projects is an accurate and reliable cost estimation process for projects and portfolios providing both cost and effort estimates as well as uncertainty predictions and risk assessment.

Effort estimates, often combined with predictions of resource utilization, are vitally important for another problem arising in this context that is tightly coupled with project selection, namely resource allocation and scheduling. A vast amount of literature deals with this problem [GT97, HVK99]. Engwall and Jerbrant [EJ03] examine the organizational setting with simultaneous projects. Shackelford and Corne [SC01] describe a collaborative project scheduling system where the search is guided both by standard schedule quality criteria and non-formalized knowledge and experience.

- *Bidding.* The core problem of bidding is simple: a bid that is too low will result in financial loss to the company if actual costs exceed the contractual price, while too high a bid will not win the project, even if it would have been profitable for the company. Thus, an accurate cost estimate is the base of every bidding decision.

  A number of articles dealing with this topic can be found in literature. Cassaigne et al. [CKSW97] present a decision support system for bidding processes developed in an ESPRIT project named DECIDE. Costing and estimation is at the heart of their proposed process model. Kitchenham et al. [KPLJ03, PKLJ02] present a model for software bidding risks. They state: "Although this paper is concerned with the bidding for software projects, its origins lie in the field of software cost estimation." and add that cost estimation processes have to be integrated into the management processes.

- *Project prioritization.* This process shows similarities to bidding as incorrect prioritization caused by inaccurate cost estimates may have fatal consequences for a company's business perspectives. Valuable resources and time may be lost due to favoring certain projects over potentially more valuable ones. Prioritization often uses concepts similar to those presented above and is also often tightly coupled with project selection. It heavily relies on well-founded estimates to assess a project portfolio's risks, costs, and benefits.

Tan and Gong [TG97] present a decision support system for R&D project selection and prioritization. According to Lukesch [Luk00], project prioritization is the core task for project selection. Again, risk-reward diagrams and variations thereof are a method widely applied.

- *Research vs. development.* It is important to find an optimally balanced portfolio that consists of both research projects exploring new business opportunities (thus often aimed towards innovation), and development projects that turn selected business opportunities into realities. It can be argued that one can see this problem as a more specific kind of project selection. Clearly, only a fraction of potentially successful research projects can normally be realized in practice. Selecting an optimal mix of projects that optimizes the total potential and value of a company's project portfolio is widely recognized as a key success factor in many modern environments. On the one hand, conventional projects creating cash flow are of obvious benefit, but on the other hand, the risk of staying behind competitors that have invested into innovative research projects cannot be ignored. Equally, investing too much in fundamental research may dry out the cash flow.

  Ernst shows the application of patent portfolios for strategic R&D planning purposes [Ern97] and develops an integrated portfolio concept for market oriented R&D planning [ES03]. Lieb [Lie98] presents a model to determine an optimal number of research projects to be developed. The research and development is viewed as a two-stage process, where the task of research is to reduce uncertainty for potential development projects. He states that there usually exists an optimal percentage of projects that should be realized and that this "fraction

> is critically dependent on the relative average research project cost and effectiveness compared to development." Obviously, without effective cost estimation and measurement, a successful balancing process is impossible.

Yet estimation of costs and efforts is notoriously difficult, for it faces a host of obstacles. First and foremost, software development most often operates in fields outside the software domain, like health care or finance—this gap results in a difficult requirement elicitation and definition, and it causes estimators to operate on ill-defined system specifications. Second, often software development can reuse past working solutions—this desired approach increases productivity and stability, yet, paradoxically, it causes large parts of the remaining and relevant development effort to be new and unknown, and more difficult to estimate.

These are main differences between software development projects and projects from other domains. However, the difficulties in estimation caused by those general software development issues are complemented by many problems specific to estimation: stakeholders often under- or overestimate projects to obtain favorable, locally optimal situations; unrealistic project targets are set to motivate or push employees; small islands of knowledge within companies create information monopolies, making estimates difficult to verify.

Fundamental to these problems are operational questions of estimation:

- Is an estimate proposal plausible and repeatable? Is the model transparent to all stakeholders? Do they accept the estimation model?

- How is estimation quality measured? How accurate and reliable is an estimation model in a given environment? What models achieve the highest estimation accuracy?

- What portfolio attributes should be measured, an how should the information be collected in typically heterogeneous environments? How much does this cost? How is the quality of historical project data?

Only satisfactory responses to these questions ensure that formal estimation methods are accepted and applied in practice. This thesis thus focuses on these operational issues; it does this by optimizing one particular brand of estimation method, the analogy-based cost estimation. Analogy-based cost estimation is a straightforward model, its estimates are based on cost information of suitable past projects—i.e., information of *similar* past projects.

In turn, several areas of potential operational improvement are identified:

- *Explicability.* Estimates are finally checked and cleared by human estimators, no matter how sophisticated the estimation method. Yet humans have difficulties in analyzing high-dimensional data sets; project similarities are difficult to assess, as they comprise many different project features. It is therefore vital, that this information be represented in a more accessible way to the estimators, allowing for fast, intuitive understanding of the data.

- *Effectiveness.* At the end of the day, estimates' accuracy and reliability are of paramount importance. Yet many methods perform weak when calibrated with insufficient data, or when parameter settings are sub-optimal. Especially the greatly varying importance or weight of different project attributes influences the estimates, but often this is delegated to "experts" or "experience" rather than dealt with. Instead, the importance or weight of the different project features should be analyzed using estimation quality criteria, both optimizing estimation accuracy, and relieving experts from making difficult feature valuations.

- *Efficiency.* Even very transparent, and highly accurate models will not be implemented in practice if the economic effort for this is perceived as too high. Defining metrics, measuring software, and collecting data is notoriously difficult to sustain over the long term. This can't be replaced completely, however, alternative ways of collecting metrics should be explored—metrics that can be derived from process artifacts instead of explicitly defining and collecting them.

For each of these areas, distinct approaches are proposed and evaluated in this thesis. In order to increase the explicability of estimates, a visualization method based on multi-dimensional scaling (MDS) is explored. This method visualizes high-dimensional data in two or three dimensions, allowing to easily grasp degrees of similarity between the visualized entities, in our case, software projects described by many project feature dimensions.

To increase the effectiveness of analogy-based estimation, extensive feature weighting is proposed. With this, optimal feature weights can be determined, with regard to common estimation quality metrics like the mean magnitude of relative error.

Finally, to improve estimation efficiency, alternative ways of assessing project or artifact similarity are explored. Using self-organizing maps (SOM), a ways to cluster high-dimensional data sets, textual similarities of requirement documents are used to cluster similar text documents using their implicit attributes, instead of explicitly collected metrics.

In academic environments, estimation methods often don't have to deal with practical annoyances like data outliers, neglected format guidelines, sparse data, etc. Thus, all three approaches were validated against real-world industrial data sets.

Quantifying the improvements was another major goal: both the approximation quality of the MDS visualization, as well as the accuracy and reliability improvements of the feature weighting are quantified using commonly used metrics.

Chapter 2

Context of Cost Estimation

This chapter presents the context of cost estimation and—in particular—software cost estimation. It describes recurring problems of software cost estimation, the estimation process, and current models and techniques to support estimation.

2.1 Estimation at Organizational Levels

This section gives an overview of the environment and organizational levels where estimation takes place and points out distinguishing aspects.

Cost estimation is at the heart of many vital project portfolio decisions, like the allocation of resources or the scheduling of project starts [Jon98]. Accurate estimates are crucial in bidding processes, where bids higher than those of competitors will result in loosing a bid, while too optimistic estimates might make a timely project execution impossible. Risk management in a project environment depends to a large degree on the assessment of the possible risk impact [Boe90]. Cost estimation affects human resources as well as pricing decisions. Cost estimation thus is highly crucial to a project portfolio's performance.

As costs are a fundamental aspect in all project environments, and as costs are a central measure of accountability, its estimates become subject of intense scrutiny by a variety of stakeholders involved [CI99]. Loosely, these diverging interests in cost estimates can be structured into three levels: the inter-company, the intra-company, and the operational level (see table 2.1, figure 2.1).

- At the *inter-company level*, estimates are often used to drive a company's business policies. A main goal can be to win bids or contracts. Here, "official" or "external" estimates can be too high (if there are few competitors, or even none), or they can be lower than "internal" estimates, i.e., the costs actually expected (for example, if it is regarded as unlikely that cost overruns result in a break of the contract or even in penalties for breach of contract). Estimates are also used to impress potential investors, or even to stall certain projects by too pessimistic estimates.

  Usually, this takes place between different companies or public institutions, or between different departments or organizational entities of large companies. A recent example of such high-level negotiations based on estimates is a large integrated system for collecting toll on German highways—it involves many political aspects: the bidding process was apparently tailored to make a German-led consortium of companies (including Deutsche Telekom and DaimlerChrysler) win the contract[1]; there was a huge schedule-overrun; the companies involved lacked a clear definition of responsibility; fierce re-negotiations failed to bring an agreement of penalties in case of further delays; finally, on February 17, 2004, the contract was canceled.[2]

  A main aspect of the involved stakeholders' behavior is the high degree of asymmetric information, i.e., the involved parties have

[1]`http://www.spiegel.de/wirtschaft/0,1518,285661,00.html`
[2]`http://www.spiegel.de/wirtschaft/0,1518,286754,00.html`

Level	**Stakeholders**	**Entities**	**Typical activities**	**Influence on estimation**
Inter-company	Chief executive officers, sales managers	Companies, governmental organizations, large departments	Bidding, tendering, consulting, funding	Underestimation can facilitate the cost justification of an unworthy project [Eme71]; too high bids may cause useful projects not to be funded [JRW00, LCB98]; business-specific requirements—like high quality in space/aviation software—may override meeting estimates [HHA91]
Intra-company	Project managers, portfolio managers, developers	Departments, team, individuals	Allocating resources, managing risks, scheduling project starts	It may be reasonable for software managers to manipulate predictions to keep it in line with targets, while developers may want to inflate predictions for easier-to-achieve targets [Hug96]; management can cause pressure via estimates and targets [MJ03]; software managers may have a tendency to over-report causes that lies outside their responsibility, e.g., customer-related causes [MJ03]
Operational	Process manager, quality assurer	Individuals and artifacts	Collecting and analyzing measurement data, creating data-driven estimates	Lack of transparent cost model can impede acceptance of formal methods [GM97]; high cost of collecting consistent portfolio measurement data causes many companies to avoid measurement programs [RJW03]

Table 2.1: Cost estimation levels: stakeholders and activities at inter-, intra-company, and operational level

a different quality of insight into the involved domains. Valid methods to investigate related research questions can involve auction analysis, or game theory.

- At the *intra-company level*, conflicting goals of individuals—often in the same company—lead to difficulties in estimation. In typical bottom-up estimation scenarios, single project members should contribute partial estimates for their area of expertise; however, it might not be in their interest to give an accurate estimate, but instead a higher one that gives some breathing space. Many are also reluctant and insecure about estimates, and try to avoid being blamed for wrong estimates—the well-known phenomenon of "taking all the available time" inevitably results. Information concentrated in small islands or even one single person, as well as the level of expertise, the risk consciousness, or personal ambitions—many aspects influence cost-related statements and estimates.

  On the other hand, top-down estimates are very often used tactically, in order to put pressure on project members, or to give the impression of a tight schedule, while at the same time hiding some tolerance or safety time range. Furthermore, high estimates can have unusually positive consequences for some stakeholders in an intra-company environment: they can actually "win" in the process of resource allocation. Questions arising at this level could be addressed using methods of behavioral economics.

  An example of a recurring phenomenon at this level is the procedure of assigning past work efforts, which have already be recorded in the work history database, to entirely different cost positions. This can happen, for example, if there is some time budget left in some projects, while others struggle with their schedule.

- Finally, at the *operational level*, estimators aim at obtaining the most accurate estimate, regardless of high-level considerations or stakeholder-specific goals. Here, informal, experience-based "guessing", in addition to formal models and methods are used to obtain fine-grained estimates. Main difficulties are incomplete requirement specifications at estimation time, or projects with a high degree of innovation—here, few historical data is available to sanity-check the estimates. Often the scope and complexity of a project is difficult to assess; many times scope and cost are not correlated in a transparent way.

The processes at this operational level yield raw internal estimates and thus provide input information for estimation decisions at the higher levels, the intra- and inter-company levels.

Of course, in order to assess the risk of making too low a bid in a bidding process between companies, or to request more resources within a company than actually needed, an accurate estimate is nevertheless vital.

Therefore, the operational level of cost estimation is fundamental to cost estimation; it provides the necessary baseline upon which all further assessments—and corresponding decisions—must rely. *Thus, this thesis focuses on the fundamental, the operational cost estimation level.*

The following sections are structured as follows:

Section 2.2 points out those aspects that distinguish software cost estimation from estimation procedures in other project environments.

Section 2.3 describes the cost estimation process, and points out the process steps at which this thesis tries to solve the problems described above: measurement data collection, estimate creation, and estimate evaluation.

Section 2.4 gives an overview of recurring problems in software cost estimation. It attributes the problems to the organizational levels, and thus points out the problems addressed in this thesis, namely, the problems at the operational level: too inaccurate, too intranspar-

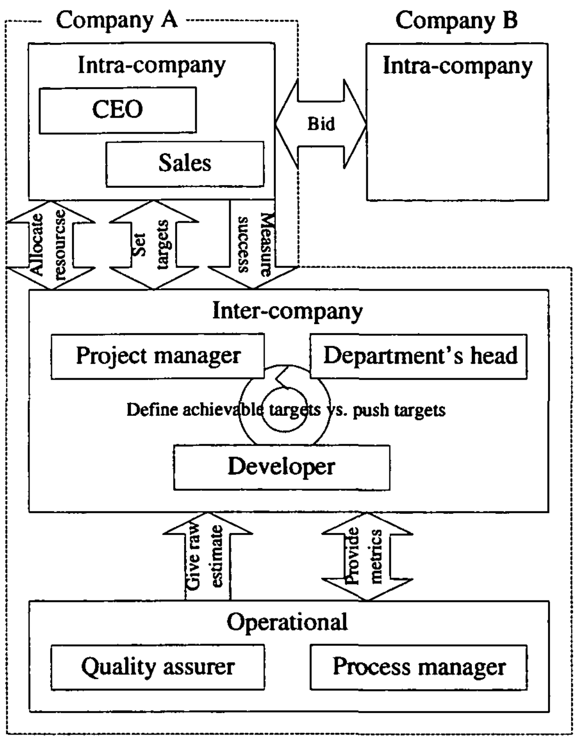

Figure 2.1: Estimation levels and stakeholders

ent, and too expensive estimates.

Section 2.5 lists common methods and techniques to support cost estimation at the operational level, and presents empirical studies on their respective strengths and weaknesses.

Finally, section 2.6 sums up these aspects; the goals are outlined in detail in the following chapter.

2.2 Software-Specific Aspects of Cost Estimation

Many software projects fail or have huge schedule and budget overruns; please refer to section 2.4 for some telling statistics. Software development seemingly differs from other engineering practices—most notably the hardware engineering environment—as being less predictable and more failure-prone.

Some authors argue that the discipline is still young and that the best practices didn't have time to evolve, whereas other disciplines (for example, architecture) had thousands of years to develop [Som94, ZBGK01]. This view—however simple—is insufficient. Modern tools, communication methods, education systems and huge economic interests should help to master an engineering discipline far faster than centuries ago. Moreover, other disciplines, like hardware engineering, seem to fit traditional quality expectations far better, even if roughly originating from the same period.

There are several software-specific aspects to the engineering process that make it difficult to master and estimate; these aspects should help to avoid unfounded expectations in formal methods.

- *Another domain's requirements.* Some types of software are aimed at operating in a software environment (like compilers), or model scientific problems (like numerical frameworks). Most other software types, for example, almost all custom database applications, must model domains that are different and distant from the software domain: health care, banking, government, etc. The corresponding business models or rules that have to be modeled with software systems do usually not follow simple mathematical mechanisms, but instead complex ones, evolved over time, based on highly irregular laws or domain-specific practices; this makes it difficult to collect requirements and to verify their consistency. Several authors thus emphasize the importance of requirement engineering [SS97b], or propose ways to ease requirement elicitation [Coc00, BBHL94].

- *Complexity.* Non-engineering approaches like ad-hoc development methods are still used widely; however, this might be viewed not just as a reason for problems, but also as a consequence of more fundamental problems in information and knowledge management—Gibbs [Gib94], for example, notes that "traditional development processes break down" when "no manager can comprehend the entirety" of a project. Current approaches to tackle this issue involve component-based systems with reliable standard software modules and components [HC01].

- *New development.* A large part of many software projects deals with previously unknown technologies and/or requirements [Wie01], especially as those parts that are already well-known can be re-used and thus represent a relatively small part of the overall development effort. For the corresponding estimation process, this leads to "inherent inaccuracy" [SA01].

- *The human factor.* The importance of the people's influence is stressed throughout the literature on software engineering. Grady Booch calls it "one of the dirty little secrets of software engineering": success in software development depends most upon the quality of the people involved [Web96]. DeMarco [DL99] states that "the major problems of our work are not so much *technological* than *sociological* in nature."

These circumstances limit the use of formal methods in the software development process to some extent; they should limit expectations with regards to a specific formal method's benefit. *Thus, this thesis regards formal methods in software cost estimation as a valuable support to the estimation process, but not as perfect, stand-alone prediction aid.*

2.3 Software Cost Estimation Process

This section will describe a process view on the operational level of software cost estimation and refer to related work at each process step.

Agarwal et al. [AKY$^+$01] describe a software cost estimation process at the operational level, consisting of the following process steps:

1. Estimate size
2. Estimate cost and effort
3. Estimate schedule
4. Estimate critical computer resources
5. Assess risks
6. Inspect/approve estimate
7. Track and report estimate
8. Measure and improve the process

Measurement data is stored in an organization software process database. In addition, several feedback mechanisms are proposed, especially between the inspection and initial estimations steps, and between the tracking and improvement steps to the process database.

Other process views are proposed, for example, by Boehm [Boe81]. His estimation process consists of the following steps: establish cost-estimating objectives, generate a project plan, pin down software requirements, work out details about the software system, use several independent cost estimation techniques to capitalize on their combined strength, compare different estimates, and monitor actual cost after project start. Yet another process view is given by Jones [Jon98].

The process can be extended by including an estimate creation step—typically involving formal methods and tools—, and simplified by grouping similar activities together, essentially yielding a 3-step estimation process with 2 feedback loops (see figure 1.2).

1. *Collection of project portfolio measurement data.* This process step can be implemented differently, according to a company's estimation processes. Ideally, company-specific portfolio data should be collected in a consistent way over a large period of time to obtain high-quality historic project information. Several studies point out that the use of such intra-company data yields significantly better results than generic data sets; Maxwell et al. [MvW96], for example, point out, that "organizational differences account for most of the productivity variation of projects."

   Often, data is not collected explicitly, but merely adds to single estimators' experience; expert judgment based on such informal data and experience handling is indeed—according, for example, to the study of Moløkken and Jørgensen [MJ03]—widespread. Without a well-defined, central measurement data repository, however, the important repeatability of estimates (demanded, for example by Briand et al. [BEB98]) is impossible to obtain.

   Several tools and framework have been proposed to facilitate the efficient collection of portfolio data. Torii and Matsumoto [TM96] describe a measurement support system for automatic process and product measurement data collection. Auer [Aue02] proposes open data formats and protocols to exchange data and gives an overview on other data collection frameworks.

   Maxwell [Max01] gives an overview on important issues when collecting software measurement data for estimation and benchmarking purpose.

2. *Creation of estimate proposal.* A variety of different approaches to cost estimate creation has been proposed. Algorithmic mod-

els like COCOMO 2 have been proposed by Boehm [Boe81]. Neural networks are used by Boetticher [Boe01], Serluca [Ser95], and Bode [Bod98]. Briand and Basili [BB92] proposed pattern recognition techniques. Other methods rely on regression analysis (e.g., Schroeder et al. [SSS86]), decision trees (e.g., Breiman et al. [BFOS84b]), or checklists and group discussion (e.g., Passing and Shepperd [PS03]).

Several studies compare the different approaches' performance. Kemerer [Kem87] reports potentially high error rates for COCOMO of up to 600 percent. Briand et al. [BLW00] compare different cost estimation techniques. Wieczorek and Ruhe [WR02] have investigated the question whether multi-organizational data is of more value to software project cost estimation than company-specific data. Shepperd and Schofield [SS97a] compare analogy-based approaches to regression analysis.

For a classification of these methods, and a description of their most prominent ones, please refer to section 2.5; there, several empirical studies comparing the various approaches are summarized as well.

3. *Check and verifications by human estimators.* When comparing different cost estimation techniques, many results are obtained by quantifying the performance of different models automatically, without human intervention (see, for example, Shepperd and Schofield [SS97a]). This neglects, however, an important issue—pointed out, for example, by Stensrud and Myrtveit [SM98]—: all estimates are sanity-checked by human estimators. The presentation of the data to the estimator, as well as the transparency of the used approach becomes thus a vital step in the overall process.

   Several authors support this additional factor. Genuchten [vG91a] speaks of models as a "second opinion", a "means of communication." Walkerden and Jeffery [WJ99a] state that people can outperform tools in selecting appropriate project analogies. Gray and MacDonell [GM99] emphasize, that when analyzing software measurement data, a model's interpretability must be taken into account.

Cost estimation for a given project is not a singular activity, but an iterative process (pointed out, for example by Mendes et al. [MMC02]), where at given points in time, newly available information is used to adapt the original estimates. The potential influence on cost, however, diminishes with the advancing project, and is described in figure 2.2 (see Becker [Bec90]).

How accurate must an estimate be? The answer to this question depends on the project stage and on the decision that the estimates influence at that particular stage. At early stages of a project—when few information is available, and requirements are only sketched—, cost estimation is primarily used to get a coarse estimate, often resulting in a probable and a worst-case estimate. This helps to categorize projects, to determine the relevant clearing level, and to prioritize them. Raiffeisen Central Bank, a large Austrian financial institution, for example, uses small, 1-page text descriptions (denoted as "Steckbrief" or warrant of apprehension) of projects for a first cost estimate.

In subsequent stages, more detailed estimates are necessary; those are based on more detailed project descriptions. In the case of Raiffeisen Central Bank, so-called preliminary surveys are used for this purpose; the estimate's quality at this stage must allow for a stop-or-go decision by the upper management. Subsequent project stages—e.g., domain-centered requirement specification, technical specification, etc.—provide ever more data to fine-tune the estimate and to optimize resource allocation.

Two feedback mechanisms in the cost estimation process (see figure 1.2) ensure that both the data measurement, as well as the estimation model calibration is refined over time, as new information based on a larger set of projects becomes available. Thus, new metrics quantifying substantial cost drivers can be added, or obsolete ones removed; model parameters or project feature weights can be set to new

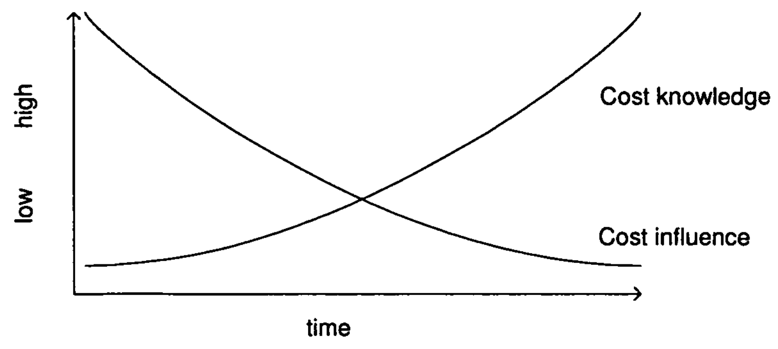

Figure 2.2: Estimation knowledge and its influence on project costs over project duration [Bec90]

values. Such calibrations are mentioned by Hughes et al. [HCYM98] as being capable of improving the model's performance. Several estimation methods, like analogy-based approaches [SS97a], can be fine-tuned at each iteration by setting the project feature weights. If more than one cost estimation model is used to create estimate proposals—in combined approaches, like proposed by Boehm [Boe81]—, the feedback can be used to validate the methods' applicability to a given environment.

2.4 Threats to Cost Estimation

This section points out reported problems related to software cost estimation, and classifies them into problem areas—according to estimation level and process step—that will be referred to when defining this thesis' goals.

Software cost estimation is vital to many prominent project portfolio decisions, like bidding, resource allocation, or start scheduling.

Nevertheless, software projects notoriously feature high mean schedule and cost overruns, and high failure rates. Moløkken and Jørgensen [MJ03] give an overview on several studies regarding estimation performance: typical mean cost overruns range from 33% to 89%; 61% to 84% of the projects are completed over budget; 65% to 84% are completed over schedule.

However, many studies indicate that the state of the practice in software cost estimation is far less sophisticated that the methods and results published in literature. Hihn [HHA91] reports, in line with results by, for example, Moløkken and Jørgensen [MJ03], that the dominant way of estimating is informal analogy, i.e., experience-based judgment. Martin [Mar98] reports that 90% of ERP implementations are late or over budget.

Many problems in cost estimation are inherent to the task of prediction, or occur at the intra- or inter-company level of organizations. Stamelos and Angelis [SA01] describe the former as "inherent inaccuracy in the estimation process" for it is a "probabilistic assessment of a future condition." Bollinger [Bol97] states that the creation of "genuinely new" software has more in common with developing a new physical theory than with an assembly line.

Many problems arise at the higher level of estimation, where decisions are influenced by many other factors than raw cost information. As Keen [Kee81] notes, computer specialists tend to over-estimate the degree to which managerial decision-making is based on information. He describe how negotiation, habit, rules of thumb, "muddling through", and information reduction under pressure often dominate decision processes.

Hughes [Hug96] describes how it may be reasonable and in the interest of managers to manipulate the prediction to keep it in line with a target, while on the other hand developers may want to inflate the prediction to make targets or deadlines easier to meet.

Moløkken and Jørgensen [MJ03] report that management puts pressure on development via cost estimates; in turn, software managers over-emphasize cost problems arising in the domain of the customer, as these are perceived as outside of their own responsibility.

Lederer and Prasas [LP96] call for a higher accountability of the stakeholders involved—estimators, developers, and managers—via performance reviews; this appears to reduce the estimation error.

Hihn [HHA91] hints at yet another reason for varying estimation performance: in specific environments, like NASA's Jet Propulsion Laboratory, software quality can be regarded as far more important than delivery on schedule—such goals or preferences can cause correct estimation and predictable resource allocation to be regarded as less important.

Other reasons for problems in cost estimation are well-known in the area of software measurement, as reported, for example, by Grady [Gra92]: insufficient management support, and difficulties in measuring productivity metrics of individuals due to privacy and motivation concerns.

Genuchten [vG91b] supports the view of a high degree of organizational influence on estimation errors; the distribution of the causes given contain more than 40% of organization-related estimation error causes.

Several authors list a mix of high-level and low-level, operational problems in the context of cost estimation. Bode [Bod98] mentions degree of uncertainty and managerial factors, but mentions additional operational problems, such as little explicit cause-effect knowledge, and little similarity between tasks.

Kitchenham and Linkman [KL97] give four reasons of estimate uncertainty, mostly regarding the operational estimation level: model error, measurement error, assumption error, and scope error.

This thesis' focus lies on the operational level of software cost estimation. The remaining part of this section refers to problems related to this level and roughly classifies them according to the respective estimation process step, either measurement data collection, estimate creation, or sanity-check and validation.

1. *Collection.* Many authors mention the notorious difficulty of

collecting valid software measurement data. Maxwell [Max01] describes several common pitfalls in software measurement. Jeffery et al. [JRW00] describes data collection as an "expensive and time-consuming process for individual organizations." Auer et al. [AGB03b] point out that the necessary automation of data collection can be difficult to achieve in highly heterogeneous software development environments.

Ruhe et al. [RJW03] declares that as data collection has to be carefully planned and supported over a long period of time, many organizations simply "do not have enough resources for the required measurement methods."

Some models thus try to avoid collecting data, and instead rely on external information. However, models based on data from different environments are reported to yield far less accurate estimates than those relying on intra-company data (see, for example, Jeffery et al. [JRW00]).

To sum up, the mere process step of measurement data collection is expensive and can—if not defined and implemented carefully—severely hinder cost estimation quality.

2. *Creation.* Creating the estimate proposal using formal or informal methods yields a variety of problematic issues. Gray and MacDonell [GM97] point out several problems of current estimation techniques, which have to deal with missing data, non-linear relationships between variables, outliers, small data set size, etc. Building a good cost estimation model is often impossible, according to Ruhe et al. [RJW03], because of the "lack of sufficient, explicit past project data to systematically build a cost estimation model." Briand et al. [BB92] mention the difficulty of assuming functional relationships, which limit classical statistic approaches, as well as non-precise or even missing data as affecting a model's validity.

Most estimation models need to be calibrated in order to achieve better estimation accuracy (for example, the analogy-based tool ANGEL [SS97a] allows to set project feature weights interactively)—alas, this requires additional effort and insight into the data.

Above all, many estimation methods exhibit a wide range of estimation accuracy and reliability. Several studies (for example, by Jeffery et al. [JRW00], or by Kemerer [Kem87]) report that a common measure of estimation quality, the mean magnitude of relative error (MMRE), is—with values of more than 100%—commonly far higher than the usually accepted 25% (please refer to Conte [CDS86]).

To sum up, it remains difficult to calibrate cost models and to create accurate and reliable cost estimate proposals, as intimate knowledge of the metrics and the estimation model is required.

3. *Check.* The final step in estimation is a verification and sanity-check by a human estimator (as pointed out by Stensrud and Myrtveit [SM98]).

   Gray and MacDonell [GM97] point out that the acceptability of a model to a human estimator is essential, including the issue of "the model 'explaining' its predictions." Ruhe et al. [RJW03] underline the lack of transparency in many software cost models, arising from their very nature as black box methods—this makes calibration by humans more difficult.

   Shepperd and Schofield [SS97a] argue that some cost estimation techniques—like neural networks—give little insight into how an estimate is produced, which makes it more difficult to re-enact an estimate, or to gain understanding of the relations of the variables. Samson et al. [SED97] express similar reservations, when saying that neural models "are of little use in explaining relationships."

Often, as with mere expert-judgment estimation (please refer to Gray et al. [GMS99]), visibility and repeatability are difficult if not impossible to achieve.

To sum up, human estimators often have to deal with intransparent estimation models and correspondingly with difficult-to-re-enact estimates; thus, model acceptance and application is impaired.

2.5 Estimation Methods and Empirical Studies

This section presents and classifies the most important alternative cost estimation methodologies.

2.5.1 Classification

Numerous attempts have been made to classify methods and models for cost estimation.

One of the first to categorize previous models was Barry Boehm [Boe81], who roughly partitions software estimation models into seven categories: algorithmic models, top-down, bottom-up, analogy, expert judgment, Parkinson, price-to-win. Unfortunately, some of these classes are not disjoint, e.g., expert judgment and analogy, but also expert judgment and bottom-up/top-down methods. And it is doubtful whether price-to-win should be considered an estimation model.

Walkderden and Jeffery [WJ99a] propose four classes of methods: empirical, analogical, theoretical, and heuristic. They exclude expert judgment from their classification; as this is the most frequently applied technique in industrial practice [HHA91, MJ03], the classification scheme must be considered incomplete. Moreover, as Wieczorek [Wie01] states, analogy seems not to be orthogonal to heuristic and empirical models.

Kitchenham [FP96] categorizes cost estimation methods into four classes: expert opinion, models, analogy, and decomposition. The latter is meant as estimating bottom-up, which unfortunately overlaps with other methods like expert judgment.

Wieczorek [Wie01] proposes a hierarchical classification scheme: model based and non-model based methods, as well as generic and specific models. Generic models further split into proprietary (ESTIMACS [Rub85]) and non-proprietary (COCOMO [Boe81]), while specific model based methods are data driven (neural networks [FW96, Bod98]) or composite (COBRA [BEB98], analogy [Bod98, SSK96]). Again, several inconsistencies can be observed. For example, the classification of analogy-based methods seems quite arbitrary; moreover, analogy appears in two distinct categories.

Several authors rely on a simpler scheme distinguishing between algorithmic and non-algorithmic models.

1. *Algorithmic models.* With varying levels of statistical sophistication, algorithmic models rely on techniques that range from regression models to differential equations [Put78]. They often rely on the concept of *productivity factors* to adjust a base estimate measured in terms of code or feature size.

   Examples include COCOMO, COBRA, and regression-based models like ordinary least squares regression (OLS), robust regression (RR), classification and regression trees (CART [BFOS84a]), etc. As opposed to other authors who list analogy-based methods as a non-algorithmic model, we refer to these as algorithmic, too.

2. *Non-algorithmic models.* The most prominent member of this category is expert judgment, being the most widely applied

technique in industrial practice. Some authors also list Parkinson, price-to-win, and top-down/bottom-up methods in this section.

As mentioned above, this scheme is not fully satisfactory either. For example, COBRA combines expert judgment to elicit knowledge about productivity factors with an algorithmic model.

As Kitchenham [KN90] already states, classification schemes are subjective and disagreement about them is common. Most researchers identify some main classes that are common, e.g., analogy, but as methods evolve and techniques are combined, a rigid classification scheme partitioning existing and emerging models into disjoint classes does not seem to be appropriate.

In the following sections, several of the main estimation methods are outlined. Note, that analogy-based cost estimation—the method improved in this thesis—will be described in more detail in section 5.3.

2.5.2 COCOMO

COCOMO stands for Constructive Cost Model. It was originally developed in 1981 by B. Boehm [Boe81] and was extended in 1995 to COCOMO II [BHM+00].

COCOMO was created to support financial decisions involving a software development process.

Overview

After the first version of COCOMO was published in 1981, it became very popular within a short period of time. COCOMO was used by many companies in the software industry and in the course of time the developer of COCOMO decided to use the mass of feedback they got to improve COCOMO. Work on COCOMO II took more than a decade. Modern trends in software engineering, and the experience with COCOMO in practice influenced the model. Another new idea of COCOMO II was to make it suitable for current and future software projects and even let companies easily adapt COCOMO II to their special situations. An important target of COCOMO II was the clear and easy-to-interpret definition of inputs and results to remove mistakes based on different model interpretations. A main goal was to allow for the calibration of COCOMO II to environments with large projects.

Description

The basic idea is to make a relation between project size and development effort; the basic relation between effort and size is given by

$$\mathit{effort} = a * \mathit{size}^b \tag{2.1}$$

The effort is given in person months and the size is specified by lines of code (LOC). The values of the variables a and b depend on the complexity of the project. COCOMO defines three kinds of project-types: organic projects (simple projects), semi-detached projects and embedded projects (complex projects).

COCOMO refines the relation between effort and project size and integrates so called "effort multipliers" (EM). These effort multipliers adjust the formula as follows

$$\mathit{effort} = a * (\mathit{size})^b * \Pi_{i=1}^{17} EM_i \tag{2.2}$$

There are several types of effort multipliers: product attributes, platform attributes, personnel attributes, and project attributes.

Product attributes describe the requirements and characteristics of the project and include

- Requirement software reliability (RELY)
- Database size (DATA)
- Documentations match to life-cycle needs (DOCU)
- Product complexity (CPLX)
- Required reusability (RUSE)

Platform attributes integrate the hardware environment and operating system which is used to run the project.

- Execution time constraint (TIME)
- Main storage constraint (STOR)
- Plattform volatility (PVOL)

Personnel attributes are the general professional abilities of the personnel working on the project, programming abilities, experience with the development environment as well as familiarity with the project's domain.

- Analyst capabilities (ACAP)
- Applications experience (APEX)
- Programmer capabilities (PCAP)
- Platform experience (PLEX)
- Programming language experience (LTEX)

- Personnel continuity (PCOM)

Project attributes comprise the constraints and conditions under which project development takes place.

- Use of software tools (TOOL)
- Multisite development (SITE)
- Requirement Development Schedule (SCED)

Finally, the former variable b is replaced by 5 "scale factors" (SF) as shown in the following formula.

$$effort = a * (size)^{1.01+0.01\sum_{j=1}^{5} SF_j} * \Pi_{i=1}^{17} EM_i \tag{2.3}$$

The scale factors describe the characteristics and quality of the software-development team or organization in 5 different areas: precedentedness (PREC), development flexibility (FLEX), architecture and risk resolution (RESL), team cohesion (TEAM) and process maturity (PMAT). Every area factor can have one of 6 values: VL (very low), L (low), N (normal), H (high), VH (very high), XH (extra high). A factor's value is determined by a project's characteristics.

Problems

As mentioned by Kitchenham and Taylor [KT85] COCOMO cannot be used randomly in any environment when the specific form of the cost relationships are not appropriate.

In addition, the model relies heavily on the concept of software size, an often-criticised measure of program complexity.

Cost Driver	**VL**	**L**	**N**	**H**	**VH**	**XH**
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally unfamiliar	largely familiar	thoroughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5

Table 2.2: Scale factor documentation [BHM+00]

Abbr.	**VL**	**L**	**N**	**H**	**VH**	**XH**
PREC	6,20	4,96	3,72	2,48	1,24	0,00
FLEX	5,07	4,05	3,04	2,03	1,01	0,00
RESL	7,07	5,65	4,24	2,83	1,41	0,00
TEAM	5,48	4,38	3,29	2,19	1,10	0,00
PMAT	7,80	6,24	4,68	3,12	1,56	0,00

Table 2.3: Scale factor values [BHM+00]

Empirical Studies

Wieczorek [Wie01] gives an overview of empirical studies on cost estimation. In her summary, 13 of 26 authors refer to COCOMO.

Here are some sample studies that compare COCOMO to other cost estimation methods.

- Chatzoglou and Macaulay [CM96] compare COCOMO to their own method MARCS (management of the requirement capture stage) and regression analysis. The study indicates that MARCS is better than regression analysis; COCOMO provides the worst results.
- Srinivasan and Fisher [SF95] compare 5 methods and COCOMO reached the penultimate position. Only the results of a method called SLIM were worse than those of COCOMO.
- Bisio and Malabocchia [BM95] compare COCOMO with their own analogy-based tool, relying on the COCOMO database for their test; they report better results for the analogy-based tool.

In none of the studies COCOMO provides the best results and in 7 of 13 cases COCOMO provides the worst results.

Summary

COCOMO II works on different levels of granularity. An essential feature of COCOMO II is the integration of several cost-drivers which makes the estimation very precise in comparison to earlier versions of COCOMO. COCOMO II uses 17 effort multipliers and 5 scale factors instead of one respectively two single values to adjust the estimates.

2.5.3 Expert Judgment

Expert judgment is a method of software estimation that is based on consultation with one or more experts. Or, in the words of one of the most important scientists in the field of software cost estimation:

"Expert judgment techniques involve consulting with one or more experts, who use their experience and understanding of the proposed project to arrive at an estimate of its cost." (Boehm [Boe81]).

Overview

Today, expert judgment is one of the most commonly used estimation methods. This popularity is caused by the following main advantages of this estimation technique:

- requires few resource in terms of time and cost;
- works with no or inappropriate data;
- yields results fast;
- may be more "intuitive" to practitioners;
- can be as accurate as other more expensive methods, especially if experts have direct experience of similar systems;
- can be adapted to exceptional circumstances.

Heemstra [Hee92] reports that 62% of estimators/organizations use expert judgment. Another study that confirmed the technique's popularity was carried out by Vidger and Kark [VK94].

But although expert judgment is, by an overwhelming majority, the most commonly used estimation method, in the software cost estimation field the majority of research work carried out has been devoted to other methods. And the few publications on this methods are mostly about combinations of expert judgment with other cost estimation techniques.

Description

Hughes [Hug96] reports that expert judgment, in its crudest form, involves the consulting of one or more experts to directly derive one resource expenditure estimate.

To estimate the effort required to complete a software project the experts should be familiar with the development environment or application domain. Usually they use their experience and understanding of a new project and available information about the new and past projects to derive an estimate. Examples of the information used are design requirements, resources available, source code, rules of thumb, software tools available, size/complexity of the new functions, data from past projects, or feedback from past estimates.

Because of the many possible causes of bias in individual experts (optimist, pessimist, desire to win, desire to please, political), it is preferable to obtain estimates from more than one expert. But if estimates are obtained from a number of experts, they inevitably differ. A very simple—but also very inefficient method—is to hold a group meeting for as long as necessary to get the experts to converge on a single estimate. Beside this method, there are many better ways to combine the different guesses into a single estimate:

- *Mean or median estimation.* The simplest and also quickest method is to compute the mean or median estimate of all the individual estimates. The main disadvantage is that this method can be distorted by one or two extreme estimates.

- *Normalized estimation.* Pfleeger [Pfl98] proposes a different approach, that is based on three predictions from every estimator:

  - a pessimistic one (a),

  - an optimistic one (b), and
  - a most likely guess (c).

  The final estimate is determined by $(a+4c+b)/6$. The variation $(a + 3 * c + b)/5$ is proposed by Steen Lichtenberg.

- *Group consensus techniques.* Beside these basic approaches, there are a lot of methods that emphazise consensus. The most popular ones are the so called Delphi method, and a refinement, the Wideband Delphi method.

Delphi Method

The Delphi method was originally developed at the RAND Corporation in 1948 for technological forecasting. The Delphi method makes use of a panel of experts, selected according to their area of expertise. Sommerville [Som94] states that "one or more experts on the software development techniques to be used and one or more experts on the application domain are consulted."

Their responses to a series of questionnaires are anonymous, and they are provided with a summary of opinions before answering the next questionnaire. So initially each expert estimates the project cost on its own by making "anonymous guesses", but the final cost estimate requires consensus. The main idea is that the group will converge toward the "best" response through an iterative process. The different guesses are used to find a group average; if a new average differs from the previous one, another round of guesses is made. In each succeeding round of questionnaires, the range of responses by the panelists will probably decrease and the median will move toward what is deemed to be the "correct" answer. This is going on until the average of the current round and the average of the prior round are identical.

One distinct advantage of the Delphi Method is that the experts never need to meet each other. This is a quite important advantage because experts are rare and their time is valuable. Another benefit is that the process does not require complete agreement by all panelists. And since the responses are anonymous, the opinions are not influenced by things like dominant positions of some participants, personalities, personal magnetism, "alleged expertise", the "pitfalls of ego"—just to mention a few of the problems reported in literature about group consensus techniques. The Delphi method attempts to avoid or at least to lessen most of these potential biases.

Wideband Delphi Method

In reviewing the results of an experiment that was performed at the Rand Corporation in 1970 to determine the relative merits of Delphi and group meeting techniques for estimating software costs, Boehm and Farquhar [Boe81] refined the standard Delphi technique. In their opinion "the written feedback did not provide a sufficiently broad communications bandwidth for the participants to exchange the volume of information necessary to calibrate their estimates with those of the other participants." This conclusion led to the development of an alternative method, the so called Wideband Delphi method. This technique is almost the same as standard Delphi but involve more discussion between the experts, held after each iteration, and focusing on large variations in the experts' estimates.

Boehm's original formulation of the Wideband Delphi method comprised of 6 steps:

1. Planning (coordinator presents each expert with a specification and an estimation form)

2. Kickoff meeting (coordinator calls a group meeting in which the

experts discuss estimation issues with the coordinator and each other)

3. Individual preparation (experts fill out forms anonymously)
4. Estimation meeting (coordinator prepares and distributes a summary of the estimates)
5. Assembling tasks (coordinator calls a group meeting, specifically focusing on having the experts discuss points where their estimates varied widely)
6. Reviewing results & iteration (experts fill out forms, again anonymously, and Steps 4 to 6 are iterated for as many rounds as appropriate)

The strengths of this method are that it combines the advantages of free discussion with the advantages of anonymous estimation. But it means a lot of overhead (team involvement, planning) and the advantage of the DelphimMethod, that the experts never need to be brought together physically, is lost (video conferences may moderate this problem).

Problems

Expert judgment highly depends on the experts selected for the investigation, their morale, their level of knowledge about the methods and tools to apply, or their expertise in cost estimation in general. The estimation is no better than the expertise and the objectivity of the estimators. So to be successful, the expert needs to have many years of experience.

Beside this very high requirements for the estimators there are some other limitations and drawbacks:

- the reasoning is known only to the owner of the estimate;

- estimation is highly subjective;
- prone to bias: personal experience, political aims, resources, time pressure, memory recall;
- use of expert judgment is not consistent and an unstructured process;
- experts with the same information will provide different cost estimates;
- estimate reuse and modification is difficult;
- knowledge loss when experts leave the company;
- estimate depends on level of experience;
- audit trail difficult to provide.

Empirical Studies

For the last twenty years, a large number of studies were published comparing different modeling and estimation techniques.

Wieczorek [Wie01] gives a summary of the most relevant empirical studies in the cost estimation field. The majority of research work carried out has been devoted to methods other than expert judgment. This especially applies to comparative studies.

Here are short descriptions of empirical studies that compare expert judgment to other cost estimation methods.

- Walkerden and Jeffery [WJ99b] compare the analogy-based techniques ACE and ANGEL to regression analysis and expert judgment. The result show that the experts yielded the best estimates, followed by analogy and linear regression. The authors conclude that people are better than tools at selecting analogies.

- Myrtveit and Stensrud [MS99] examine expert judgment in combination with other estimation methods. Best result were obtained with the combination of expert and multiple regression followed by expert and analogy. In general, the combination of expert judgment with any formal method seems to perform better than expert judgment on its own.
- Mukhopadhyay et al. [MVP92] investigate expert judgment, function points, COCOMO and Estor (a method proposed by the authors). Expert judgment appears to be the most accurate method, whereas COCOMO is last.
- Vicinanza et al. [VMP91] compared COCOMO, function point-based regression models, and expert performance. Apparently, managers make more accurate estimates and therefore outperform function point-based regression models, which in turn outperform COCOMO.

In these four reports expert judgment yielded the best or at least very good estimates in terms of accuracy.

Jørgensen [Jør04] provides another very extensive review of empirical studies, all of them related to expert estimation of software development effort. A total of fifteen different empirical studies comparing expert estimates with estimates based on formal estimation models are presented. But the results of the studies were not conclusive, because the fifteen different empirical studies either

- are in favor of expert judgment;
- can't find a significant difference; or
- are in favor of model-based estimation.

So there is no substantial evidence in favor of either model- or expert-based estimation. Jørgensen concludes that expert estimates seem to be more accurate when there is important domain knowledge not included in the estimation models, when the estimation uncertainty is high as a result of environmental changes not included in the model, or when simple estimation strategies lead to relatively accurate estimates.

Summary

Expert judgment is one of the most commonly used estimation methods. It requires few resources in terms of time and cost, is quite intuitive to use and is applicable in nearly every circumstance. Nevertheless, it can be as accurate as other, more expensive methods, especially if experts have direct experience of similar systems.

But because of the high dependency on the experts, it must be pointed out that the estimation is no better than the expertise and the objectivity of the estimators.

2.5.4 Neural Networks

Overview

Artificial neural networks (ANN) are biologically inspired methods, based on the neural structure of the brain. Or in other words: they are a computer simulation of a "brain-like" system of interconnected processing units.

Such a network is composed of many neurons that are linked together according to a specific network architecture, with the intention to transform the inputs into meaningful outputs.

Although an ANN can model a complex set of relationships between dependent variables (e.g., effort) and independent variables (e.g., cost drivers) and furthermore allows the learning from previous situations and outcomes, it is not a very common software cost estimation practice. This is due to the fact that ANN is quite difficult to understand and "it is not possible to obtain any explanation or justification of an estimate" (Finnie and Wittig [FW97]).

Description

Artificial neural networks (ANN) belong to the class of machine-learning methods. The ANN approach is inspired from biological nerve nets and can be characterized by its

- architecture,
- activation functions, and
- learning functions.

Today, there exist a lot of different models of ANN. For example, with respect to the architecture they can be grouped in feed-forward networks (with no loops in the network path) and feedback networks (with recursive loops). In almost the same manner it is possible to distinct the different ANN regarding the activation functions or the learning functions.

Here, we will concentrate on presenting

- feed-forward multi-layer perceptron (MLP) networks,
- the Sigmoid function, and
- the back-propagation algorithm,

which Idri and Khoshgoftaar [IKA02] propose as the most commonly adopted architecture, activation function and learning algorithm for software cost estimation modeling.

Architecture

The main building elements of a neural network are the "nodes" (neurons) and the "links" connecting the different nodes. In feed-forward MLP networks the neurons are usually arranged in layers and there are only connections between adjacent layers. There are three different types of nodes:

- input nodes (e.g., cost drivers),
- output nodes (e.g., effort), and
- hidden or intermediate nodes.

Activation function

An ANN generates an output by propagating the initial inputs through "subsequent layers of processing elements to the final output layer" (Idri and Khoshgoftaar [IKA02]).

In more detail, each link in the ANN has a unique weight value. When a node receives inputs from connected nodes in the previous layer, it multiplies each of those signals by the corresponding weight value. Then the weighted inputs are summed, and passed to an activation function, which implements the "firing" of the neuron.

The favored activation function for the nodes in the hidden layer is the Sigmoid function:

$$f(x) = \frac{1}{1 + e^x} \tag{2.4}$$

For the last layer, the identity function is used and the result is not the input value for the next layer but rather the response of the network (e.g., the estimate of a software development effort).

Learning function

Before the network will be ready to make estimates for new projects, it must be "trained", what means, that the weight values (w_{ij}) must be set to values which produce correct outputs to input patterns.

The most commonly used training or learning function for ANN is back-propagation.

This function requires that all synaptic weights (w_{ij}) in the network are initially set to small, random values. The backpropagation algorithm then adjusts the weights of the connections at each node, in order to reduce the discrepancies between the network outputs before adjustment and the desired values, usually by iterating the training data several times.

Problems

While artificial neural networks can model complex relationships, there are also some limitations and drawbacks:

- "determining why an artificial neural networks makes a particular decision is a difficult task" (Idri and Khoshgoftaar [IKA02]);
- performance is sensitive to certain configuration choices, but neither ideal parameter settings nor guidelines for the construction of the neural networks topologies (number of hidden layers, number of nodes per layer, initial weights, ...) are available;
- performance is to a large degree dependent on the selected training data;
- "not possible to obtain any explanation or justification of an estimate" (Finnie and Wittig [FW97]);
- requires re-training to incorporate new data;
- input and outputs are restricted to numeric values;

- good "record keeping" for all software projects is necessary and such records must also include information on all estimates and adaptations to estimates at various stages of the project.

Empirical studies

Several publications compare ANN to other cost estimation methods.

- Srinivasan and Fisher [SF95] compare five different methods: regression trees, artificial neural networks, function points, COCOMO, and SLIM. The artificial neural networks achieved the best result, followed by FP and regression trees.
- Jørgensen [Jør95] compares several variations of regression, artificial neural networks, and combinations of OSR with regression. He found that multiple regression models and a combination of OSR with regression worked best in terms of accuracy.
- Grady and MacDonell [GM99] investigate the influence of different data set characteristics on the performance of OLS regression, robust regression, and ANN. They used different measures to evaluate performance, but in all cases OLS and robust regression performed better than ANN.
- Mair et al. [MKL$^+$00] compare ANN, case-based reasoning (CBR) and rule induction (RI). They evaluate the accuracy but also subjectively evaluate the explanatory value and configurability of the applied methods. ANN had superior accuracy, but after summarizing the relative merits and demerits, ANN is not considered the best choice for building prediction systems.

The results of the different empirical studies are not conclusive. In most studies artificial neural networks achieved quite good results in terms of accuracy. But in other studies ANN is clearly outperformed by alternative approaches, not only when other qualities than accuracy are evaluated.

Summary

An artificial neural network is a mathematical model for information processing inspired by biological neural networks.

After training, artificial neural networks have the ability to model complex non-linear relationships.

ANN achieves good results in terms of accuracy in most empirical studies. But there are some limitations and drawbacks that prevent it from being accepted as a common software cost estimation method.

2.5.5 Regression

Overview

Fitting a model to data is the common theme of a large part of statistical tasks in science and technology. The methods most often used for this task are based on the least squares (LS) principle, because it is quite simple, relatively easy and cheap to compute.

A further significant advantage of regression in general is the visibility of the resulting models, which enables an experienced modeler to immediately consider the validity of the models structure.

Description

Regression is a widely used and quite popular statistical technique for estimating resources in software engineering, where one or more independent variables are related to a dependent variable. The relation is described by the "regression line", which is the best approximation of the relationship between the dependent and the independent variables.

A simple example for resource estimation:

$$\mathit{effort} = \beta_1 + \beta_2 * \mathit{size} \tag{2.5}$$

After creating this simple regression model with data from past projects, future effort estimates can be computed on the basis of the project size. Certainly in practice there are not only one but many independent variables (cost drivers).

Today there exist a lot of different regression-based methods, but in most cases they only differ by the technique to minimize the inaccuracy of the regression line.

The following sections describe main types of regression approaches.

Ordinary least squares (OLS) method

"Ordinary least-square regression (OLS) assumes a functional form relating one dependent variable (e.g., effort), to one or more independent variables (i.e., cost drivers)" (Berry and Feldman [BF85]).

The reasons for the popularity of OLS include ease of use and simplicity. Moreover OLS is easily accessible through most statistical analysis packages, computationally cheap and it is widely examined in standard statistical texts.

In general a model using the OLS method can be written as:

$$y_t = \beta_1 + \beta_2 x_{t2} + \ldots + \beta_k x_{tk} + e_t, \tag{2.6}$$

where $x_{t2} \ldots x_{tk}$ are regressor variables for the t-th observation, $\beta_2 \ldots \beta_k$ are response coefficients, and y_t is the response variable for the t-th observation. The error term e_t is a random variable with a—typically normal—probability distribution.

The OLS method then fits the data to the specified model (i.e., finds values for $\beta_1 \ldots \beta_k$) trying to minimize the overall sum of squared errors; the main goal is to minimize the following least squares error term:

$$\sum_{i=1}^{n}(Y_i - \widehat{Y_i})^2 \tag{2.7}$$

where Y_i is the model's predicted response and $\widehat{Y_i}$ the observed response for the i-th observation.

Standard regression is well known to be optimal if the random errors are normal distributed. However, if this assumption is not valid, especially in the presence of outliers, there exist alternatives that perform better: the robust regression techniques.

Robust Regression Techniques

Robust regression is an improvement over the standard OLS approach, "because statistical techniques that fall in this category produce predictive models that are generally more effective for making predictions for the main body of observations in data sets containing outlier observations" (Gray and MacDonell [GM99]).

Some examples of statistical techniques that fall in this category of robust regression techniques are:

Least Median Squares (LMS)

The LMS method is very similar to the OLS method. The only difference is that this technique minimizes the median of the squares of absolute error instead of the sum of the squares of absolute error. This makes LMS more stable with respect to outliers.

Least Trimmed Squares (LTS)

LTS and LMS are almost the same; LTS minimizes a trimmed sum of the squares of absolute error.

Least Squares of Balanced Relative Errors (LBRS)

Instead of minimizing the sum of the squares of absolute error, the idee of LBRS is to minimize the sum of the squares of relative error (Miyazaki et al. [MTON94]).

Weighted OLS Regressions

This variant of robust regression minimizes the sum of weighted squares of absolute error, where special weight functions are used to determine the weights.

Problems

Beside the main advantages of regression-based techniques (simplicity and the wide acceptance), there are also some limitations and drawbacks:

- cannot handle missing data;

- could be difficult to interpret for non-statisticians;
- least-squares models may be extensively confounded by outlier data points, or with other words, "the regression model can be distorted by the existence of outlying observations" (Kemerer [Kem87]);
- the predictor variables must not be correlated, but most of the existing software estimation models have parameters that are correlated to each other;
- valid assumptions about the form of the functional relationships between predictors and their distributions are difficult to achieve;
- regression may lead to the provision of equations that are difficult to interpret in an operational sense, particularly when many variables, transformations, and interaction terms are included;
- regression techniques use dummy variables to deal with discrete independent variables, which increases the number of variables and therefore makes the model more difficult to interpret.

Robust regression (RR) methods can avoid some of the disadvantages, especially the problem with data sets containing outlier observations. But simultaneously RR introduce some new problems:

- because desired robustness properties can only be achieved at the cost of heavy computation, the most RR techniques are computationally quite intensive;
- usually RR methods are less powerful than standard regression methods when data set characteristics are not problematic;
- RR methods are not as widely accessible in terms of software availability;

- generally tool support is required because the RR methods are more complex than standard regression methods like OLS.

Empirical Studies

Here are some short descriptions of empirical studies in which regression-based techniques are compared to other cost estimation methods:

- Briand et al. [BB92] compare the COCOMO model, stepwise OLS regression, and optimized set reduction (OSR). In this study OSR outperforms stepwise regression, which in turn outperforms the COCOMO model.
- Subramanian and Breslawski [SB93] compare the COCOMO model to two regression models and found that the two regression models performed better in terms of MMRE (mean magnitude of relative error) than the COCOMO model.
- Shepperd and Schofield [SS97a] compare analogy to different regression models with six different data sets. In their study analogy outperforms regression in all cases.
- Walkerden and Jeffery [WJ99a] compare the analogy-based techniques ACE and ANGEL to regression analysis and expert judgment; experts yield the best estimates followed by analogy and linear regression.
- Briand et al. [BLW00] compare OLS regression, stepwise ANOVA, analogy, CART, and combinations of these techniques. The results indicate that OLS regression and ANOVA perform better than other evaluated techniques. However, consistent with previous research, even the best models are found to be inaccurate.

- Miyazaki et al. [MTON94] compare the robust regression method LBRS (least squares of balanced relative errors) with the standard regression method OLS (ordinary least squares). For all evaluation criteria the results of the OLS solution are worse than the results of the LBRS.

Summary

Regression-based techniques are widely used in software cost estimation because of their simplicity and the wide acceptance. Furthermore a lot of existing parametric cost models use some form of regression, like COCOMO II, SLIM, etc.

The best-known standard regression technique is the ordinary least squares (OLS) method, which is often called the basis of regression and econometrics.

An improvement over the standard OLS approach are the so-called robust regression techniques, which try to decrease the problem of outliers.

Although the different regression-based techniques performed considerably well in most empirical studies, there are some limitations and drawbacks for both types of regression, like the inability to deal with missing data.

2.6 Summary

This chapter gave an overview on the field of software cost estimation, and the various organizational levels where estimation takes place. It presented a process view of the operational level of cost estimation, consisting of three steps: collection of data, creation of estimate proposal, and check and verification by human estimator.

Then, the various problem areas associated to these process steps were identified. An overview was given on current cost estimation methods, along with empirical studies evaluating the different approaches' estimation performance.

Chapter 3

Goals

This chapter presents the thesis' goals depending on the problem areas identified in chapter 2. It outlines the proposed approaches and defines measures to assess whether the goals have been reached.

3.1 Overview

Chapter 2 identifies problem areas at the various stages of the software cost estimation process at the operational level (depicted in figure 1.2):

1. *Stage 1: Collect.* The mere process step of measurement data collection is expensive and can—if not defined and implemented carefully—severely hinder cost estimation quality.

2. *Stage 2: Create.* It remains difficult to calibrate cost models and to create accurate and reliable cost estimate proposals.

3. *Stage 3: Check.* Human estimators often have to deal with intransparent estimation models and correspondingly with difficult-to-re-enact estimates; thus, model acceptance and usage is low.

Each of the problem areas comprises several distinct problem types; however, some aspects stand out in each case:

- At the check step of the cost estimation process, estimators need to verify and sanity-check estimate proposals created in the creation step. While it is important that this can be performed in an efficient way, the main aspect is the transparency of the estimate creation and presentation, the *explicability.*
- At the creation step of the cost estimation process, a creation method's *effectiveness* determines estimation accuracy and reliability. Again, this step is performed relatively seldom, thus efficiency criteria are of secondary importance.
- The collection step is to be performed continuously; while it can easily be made transparent, it is difficult to assure its *efficiency.*

Thus, three main goals of this thesis can be derived:

1. Increase the explicability of estimation at the check step.
2. Increase the effectiveness of estimation at the creation step.
3. Increase the efficiency of estimation at the collection step.

The following sections describe these goals in greater details, and outline the corresponding research questions and hypotheses. Note: the approaches proposed in this thesis are in the reverse order of the cost estimation process; this is due to the order in which these partial results were published, and because subsequent approaches optimized several details of previous ones.

3.2 General Assumptions

This section lists and motivates the main assumptions on which the subsequent approaches rely.

- *Acceptance.* A key element of every estimation method must be end-user acceptance. Complicated, intransparent methods are unlikely to be used in a real-world environment, as they are difficult to learn, to re-enact, and to use when communicating with others. Black-box methods like neural networks do little to make estimators understand how an estimate was actually produced (pointed out, for example, by Shepperd and Schofield [SS97a]). Several other authors, like Gray and MacDonell [GM97], or Ruhe et al. [RJW03] stress the importance of transparency and model acceptability for human estimators.

  In line with these requirements, Hihn [HHA91] and Moløkken and Jørgensen [MJ03] report, that many of the more complicated estimation models are used very rarely.

  This thesis thus focuses on transparent models with simple-to-re-enact estimate proposals.

- *Flexible data properties.* Real-world portfolio measurement data or software metrics have several characteristics that make data processing non-trivial. As pointed out by Briand and Basili [BB92], variables can be both continuous or discrete; often data values are missing; some variables affect the model differently depending on the particular part of its value range (a phenomenon known as "heteroscedasticity"); typically, variables have a skewed distribution.

  This thesis thus applies methods with the few or no assumptions on variable distribution and scale type; it favors approaches that can easily deal with corrupt or missing data values.

- *Company-specific data.* Comparing projects and processes between different companies is notoriously difficult; too many factors—domain, business values, cultural background—greatly affect project or process instances. While it would be economically tempting to use estimation models which are already calibrated, thereby avoiding costly portfolio measurement data collection, the performance of such generic models is considerably worse than those calibrated with company-specific information (reported, for example, by Jeffery et al. [JRW00]).

This thesis thus concentrates on methods which rely on measurement data collected in the very environment where the estimation takes place.

- *Standard quality measures.* There are many studies comparing different cost estimation methods in various environments and on different data sets. However, they are often difficult to reproduce, as different quality measures are used. Sheperd and Schofield [SS97a] emphasize that using standard measures like mean magnitude of relative error yield better comparability—effectively outweighing the problems associated with these metrics by some authors (for example, Miyazaki et al. [MTN+91]).

  This thesis thus uses standard measures to quantify and compare estimation method performances.

- *Real-world data sets in public domain.* Methods should be applied to real-world data sets, which often feature missing values, outliers, etc.—but only that way a methods stability can be guaranteed. And similarly to the previous point, comparisons and studies on various data sets should be reproducible; using data not available in the public domain is a barrier for this.

  This thesis thus analyzes the proposed approaches on well-known real-world data sets available in the public domain.

- *Estimators' influence.* Estimation methods and models support estimators; however, the estimate will always be checked, verified and approved by human estimators (pointed out by, for example, by Stensrud and Myrtveit [SM98]). Therefore, criteria important to humans, like data preparation or reporting and model transparency, must be taken into account—in the worst case, a model is not accepted (see above), often, information overflow or inefficient reporting does not allow unleashing all possible information.

This thesis thus addresses data preparation issues relevant to human estimators; however, it does address them specifically and avoids measuring the aggregate quality improvement consisting of aspects relevant to estimators as well as model improvements—that way, it is easier to separate the various methods' effects.

3.3 Goals and Hypotheses

This section outlines this thesis' goals in detail and gives measures to verify whether they are met.

The goals address one particular estimation method, analogy-based cost estimation [SS97a]. For a detailed introduction to the analogy-based approach, please refer to section 5.3. This method has been reported as among the best estimation methods [Wie01]. In addition, it is one of the most straightforward methods, implementing the generally valid estimation assumption of "projects are expected to behave like similar past projects" directly, using distance functions on project features. While a large body of literature explores various aspects of the analogy-based approach, this thesis' improvements offer considerable optimizations.

3.3.1 Explicability of an Estimate

Estimates are typically created in a variant of a generic estimation process depicted in figure 1.2. The process is influenced by a variety of factors (data quality, estimators' expertise, used models, portfolio environment, etc.)—yet much research effort tries to automatically assess tools' or methods' estimation performance as measured by accuracy metrics [SS97a].

While yielding important insights, these approaches are not sufficient to achieve much-needed high quality estimation, for two reasons:

- The estimator's influence is not addressed. Every estimate must finally be approved by the decision maker—as [SM98] point out—, which greatly affects the results especially in case of outliers or unlikely estimates, where the mere automatic application of estimation tools notoriously fails.
- In addition to the often-used effectiveness criteria like accuracy and reliability, many other, secondary criteria must be addressed as well. Efficiency criteria (estimation effort, learning effort), usability (both to novices and experts), transparency of the model etc. greatly affect the acceptance of cost estimation methods and processes; if not addressed properly, decision makers will not apply the proposed approaches. [HHA91] describe how few methods are actually applied in industrial environments; [SS97a] indicate that some complex estimation methods often provide little insight on why a specific estimate is proposed, which may be a reason for their lack of acceptance.

The estimator's performance and the acceptance and transparency of the method or process are thus elementary to achieve high-quality estimates. Therefore, the presentation of the model and portfolio data to the estimator becomes fundamental. Unfortunately, people are generally not performing well at analyzing typical raw project portfolio data—high-dimensional data sets—, due to the high search effort to link the data items literally spread out in a spreadsheet format. According to [RS02], the assimilation of such information is not intuitive while visualization aids the understanding. According to [LS87], features are often more easily extracted from diagrams than from tabular or sentential representations, because some diagram types can group together related concepts, while tabular representations may store related items in separate areas, resulting in higher search efforts for linking concepts.

Standard methods to handle such high-dimensional data (like regression analysis or traditional analogy-based approaches) propose estimates, but it is difficult to understand if the result can be trusted—estimators do not know how confident they can be with an estimate proposal.

To overcome these fundamental analysis and recognition difficulties, this thesis aims at applying advanced visualization methods to project portfolio data. Multidimensional scaling methods are applied to visualize high-dimensional data in two or three dimensions; this way, the project portfolio data becomes understandable as the data is clustered visually, yielding an immediate aggregate overview of the portfolio. The visualization relies on the concept of similarity or analogy between projects, which can be expressed using similarity (or dissimilarity) values between the n projects—$n(n-1)/2$ values in the symmetric case—, or Minkowski distance functions on the projects' features, i.e., the data dimensions.

This thesis proposes an MDS-based user interface to high-dimensional project portfolio data to support software cost estimation. It applies this approach to several real-world industry project portfolio data sets and it quantifies the MDS approximation quality. Finally, it outlines promising benefits by referring to visualized project portfolio properties.

This approach should give estimators an intuitive insight into portfolio data and exploit human cognition and pattern processing, thus achieving an effective, efficient and accepted estimation method, as well as a better understanding of the correlation between data characteristics and estimation methods' accuracies.

- People can immediately assess the structure of portfolio data, especially the clusters of similar projects—this eases identification of outliers or unusual project behavior and allows for higher estimation accuracy and reliability. In addition, an estimate's confidence can easily be determined, for example, when the project to be estimated is similar to a large, dense cluster of projects performing similarly the estimator can be confident with an analogy-based estimate proposal.

- The method is visual, the mathematical model transparent, the process fast and easy-to-learn—this should guarantee high acceptance and low estimation effort. The interactive playing with the data set—i.e., choosing subsets of the data dimensions, zooming in on particular interesting project clusters—will enhance portfolio understanding and influence portfolio measurement.

More strictly, this thesis outlines expected benefits in the areas of model transparency, portfolio overview and understanding, selection of methodology, operational data handling, and estimation confidence assessment.

To gain these qualitative improvements, first it must be determined whether the approximation quality of the visualization can be regarded as high enough. Using a quantitative measure used in the field of MDS—Kruskal's stress value—, several real-world portfolio data sets are analyzed. The question is, whether MDS visualization of such sets achieve high approximation. MDS literature often proposes the high-quality classes good and excellent, corresponding to stress values lower than 0.05 or 0.025, respectively.

3.3.2 Effectiveness

A broad range of processes, procedures, mathematical methods and tools have been proposed to support cost estimation; many studies compare existing approaches—for example, regression models, expert judgment, or analogy-based methods—with respect to various estimation quality metrics, for example, the accuracy metric mean magnitude of relative error (a good overview is given in [Wie01]).

However, due to many constraints in empirical data collection—inconsistent quality of the collected software metrics, environments differing widely in domain or scope, different expertise level of estimators, software portfolio data properties due to small sample sizes, etc.—it remains difficult to generalize many of the obtained results, as pointed out by Myrtveit and Stensrud [MS99].

Nevertheless, several conclusions can be drawn safely:

- While formal estimation methods (like regression analysis or analogy-based approaches) are not the only relevant aspect of the estimation process, they do increase the estimators' performance [SM98].
- While there is no single best cost estimation method for all circumstances, analogy-based approaches rank among the best methods in a variety of environments, as pointed out by Wieczorek who gives an overview on previous study results [Wie01].
- Finally, while a method's estimation quality measures are important, other, secondary "usability" attributes—like transparency and simplicity—greatly affect a method's acceptance and applicability in practice; the analogy-based approach is both transparent and simple [SS97a, MC00].

To sum up: cost estimation is vital to software project portfolio management; cost estimation can be improved by supporting estimators with mathematical models and tools; cost estimation relying on the analogy-based approach is accepted and yields promising performance.

The traditional analogy-based approach is based on deriving estimates from historical project portfolio data by finding one or several projects (with known cost, effort, or productivity metrics) that are similar to the project to be estimated. Projects are regarded as "similar" if they feature similar project metrics or features (like number of project team members, expertise level, function point count, etc.). The degree of similarity is usually determined using the Euclidean distance on the l-dimensional feature space. However, traditional approaches do not address the features' different importance or weight adequately.

This thesis aims at extending analogy-based approaches as described by Shepperd and Schofield [SS97a], in order

1. to increase estimation accuracy as measured by the mean magnitude of relative error *MMRE* (*MMRE* describes the mean magnitude of the relative estimation error expressed in %, when estimating every project p of a portfolio P using the portfolio $P \setminus \{p\}$ as historical feature repository);

2. to increase estimation reliability as measured by the variance of the relative error Var_r (Var_r measures the error distribution, i.e., how likely estimates are to differ from the actual value);

3. to increase the model's stability as measured by the volatility of dimension weight values *MMDWC* (*MMDWC* is a measure that describes, how the optimal weight values change when new projects are added to a portfolio).

This is achieved by replacing conventional feature dimension *selection*—where subsets of features are determined for computing distances—by more flexible feature dimension *weighting*—where features are weighted differently and can thus influence the distance function with varying degrees—, and by implementing a tool for this more computation-intensive approach. Finally, we apply the approach to five industrial project portfolio data sets available in the public domain.

The resulting approach supports estimators better, as more accurate and reliable estimates are created, and as the model is more stable with respect to changes in the underlying software portfolio data, i.e., when adding new project data.

In addition, and maybe more importantly, if relying on the same basic assumptions (no outliers are removed from the portfolio; common estimation quality measures are used; the Euclidean distance is used; and merely the nearest projects are used to calculate the estimate), the new approach indicates an estimation quality barrier that can not be broken even by experienced estimators knowing about the different features' impact on the project similarities.

3.3.3 Efficiency

Most relevant software portfolio decisions—like resource allocation, bidding, or start scheduling—rely on supporting procedures like cost estimation or risk assessment. Usually, a key for successful decision support is seen in software and process measurement, i.e., the definition, collection, storage and analysis of past project features, preferably in quantitative metrics.

Yet in many industrial environments such explicit measurement programs are never implemented, or fail [RJW03]. The reasons, outlined in more detail in the following section, are manifold, but they can be summed up: project team members often perceive such programs as economically inefficient, at least in the short term.

However, in certain circumstances *explicit* measurement might be unnecessary; instead, *implicit* metrics can be used, merely derived from artifacts that are created anyway during the development process, yielding valuable information without or with negligible extra cost.

One area we deem suitable for such an approach is analogy-based cost estimation. It tries to identify past projects that are similar to the project to estimate, and to create an estimate proposal by using the historical cost data. Traditionally, analogies are defined as distances of explicitly collected project features or metrics.

This thesis proposes to use the textual description of requirement documents—more precisely, structured chunks of documents, use cases—to determine implicitly the similarity of various requirements, allowing for a fast search of similar past project requirements without explicit project metrics. This thesis evaluates this approach on several industrial use case sets, and finds out, whether those sets can be used without substantial modification in writing them.

3.4 Summary

This chapter derived the goals of this thesis based on the problem areas and their improvement potential identified in the previous chapter. The underlying assumptions were given and motivated.

Three main goals were defined, regarding the explicability (i.e., the transparency, acceptability and intuitiveness), the effectiveness (i.e., the accuracy, reliability and volatility), and the efficiency (i.e., the effort required to obtain measurement data) of the particularly well-suited analogy-based approach to cost estimation.

To achieve these goals, several steps of the overall cost estimation process are addressed: the collection of measurement data, the creation of estimate proposals, and the check by human estimators.

Quantitative measures are given to make it possible to assess whether the goals have been reached.

Chapter 4

Estimation Explicability

Software cost estimation is a crucial task in software project portfolio decisions like start scheduling, resource allocation, or bidding. A variety of estimation methods have been proposed to support estimators.

Especially the analogy-based approach—based on a project's similarities with past projects—has been reported as both efficient and relatively transparent. However, its performance was typically measured automatically and the effect of human estimators' sanity checks was neglected.

Thus, this chapter proposes the visualization of high-dimensional software project portfolio data using multidimensional scaling (MDS). We (i) propose data preparation steps for an MDS visualization of software portfolio data, (ii) visualize several real-world industry project portfolio data sets and quantify the achieved approximation quality to assess the feasibility, and (iii) outline the expected benefits referring to the visualized portfolios' properties.

This approach offers several promising benefits by enhancing portfolio data understanding and by providing intuitive means for estimators to assess an estimate's plausibility.

4.1 The Need for Transparency

Cost and effort estimation [Jon98, BHM+00, CDS86] is a ubiquitous task in software project environments, which are typically multi-project environments or *software project portfolios.* High-quality estimates are fundamental to stakeholders—success-critical project participants like project and portfolio managers, as well as quality managers—in making a variety of prominent software project portfolio decisions, for example, in the quotation phase and bidding process, in resource allocation, in project start scheduling, or in risk management. Estimation quality thus greatly affects a project portfolio's performance—*high-quality estimates are vital in making portfolio decisions.*

Estimates are typically created in a variant of a generic estimation process depicted in figure 1.2 (a more detailed process is proposed by [AKY+01]). The process is influenced by a variety of factors (data quality, estimators' expertise, used models, portfolio environment, etc.)—yet much research effort tries to automatically assess tools' or methods' estimation performance as measured by accuracy metrics [SS97a].

While yielding important insights, these approaches are not sufficient to achieve much-needed high quality estimation, for two reasons:

- The estimator's influence is not addressed. Every estimate must finally be approved by the decision maker—as [SM98] point out—, which greatly affects the results especially in case of outliers or unlikely estimates, where the mere automatic application of estimation tools notoriously fails.

- In addition to the often-used effectiveness criteria like accuracy and reliability, many other, secondary criteria must be addressed as well. Efficiency criteria (estimation effort, learning effort), usability (both to novices and experts), transparency of the model etc. greatly affect the acceptance of cost estimation methods and processes; if not addressed properly, decision mak-

> ers will not apply the proposed approaches. [HHA91] describe how few methods are actually applied in industrial environments; [SS97a] indicate that some complex estimation methods often provide little insight on why a specific estimate is proposed, which may be a reason for their lack of acceptance.

The estimator's performance and the acceptance and transparency of the method or process are thus elementary to achieve high-quality estimates. Therefore, the presentation of the model and portfolio data to the estimator becomes fundamental. Unfortunately, people are generally not performing well at analyzing typical raw project portfolio data—high-dimensional data sets—, due to the high search effort to link the data items literally spread out in a spreadsheet format. According to [RS02], the assimilation of such information is not intuitive while visualization aids the understanding. According to [LS87], features are often more easily extracted from diagrams than from tabular or sentential representations, because some diagram types can group together related concepts, while tabular representations may store related items in separate areas, resulting in higher search efforts for linking concepts.

Standard methods to handle such high-dimensional data (like regression analysis or traditional analogy-based approaches) propose estimates, but it is difficult to understand if the result can be trusted—estimators do not know how confident they can be with an estimate proposal.

To overcome these fundamental analysis and recognition difficulties, this thesis aims at applying advanced visualization methods to project portfolio data. Multidimensional scaling methods are applied to visualize high-dimensional data in two or three dimensions; this way, the project portfolio data becomes understandable as the data is clustered visually, yielding an immediate aggregate overview of the portfolio. The visualization relies on the concept of similarity or analogy between projects, which can be expressed using similarity (or dissimilarity) values between the n projects—$n(n-1)/2$ values in the

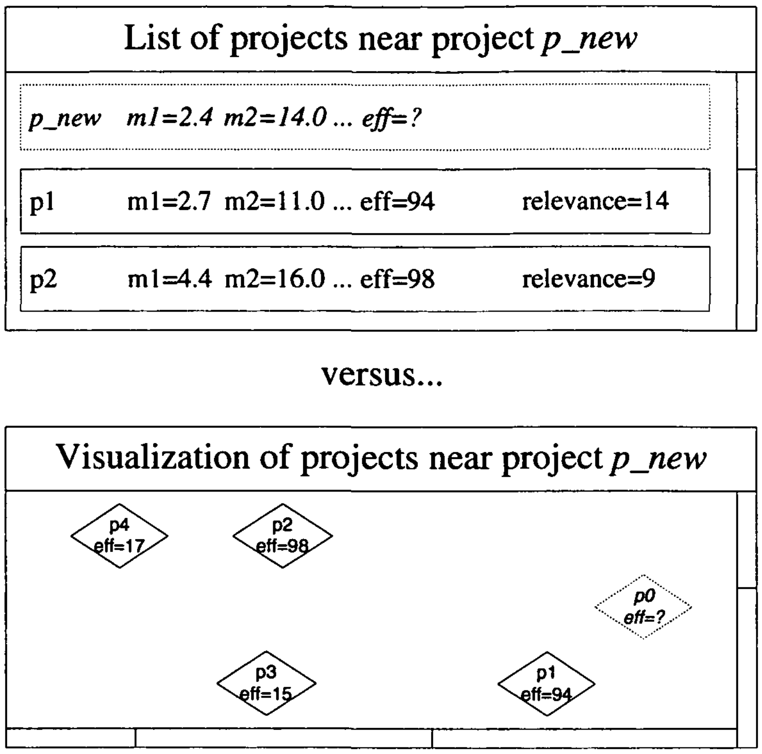
List of projects near project p_new
p_new m1=2.4 m2=14.0 ... eff=?
p1 m1=2.7 m2=11.0 ... eff=94 relevance=14
p2 m1=4.4 m2=16.0 ... eff=98 relevance=9
versus...
Visualization of projects near project p_new
p4 eff=17
p2 eff=98
p0 eff=?
p3 eff=15
p1 eff=94

Figure 4.1: Listed vs. visualized analogies

symmetric case—, or Minkowski distance functions on the projects' features, i.e., the data dimensions.

This thesis proposes an MDS-based user interface to high-dimensional project portfolio data to support software cost estimation. It applies this approach to several real-world industry project portfolio data sets and it quantifies the MDS approximation quality. Finally, it outlines promising benefits by referring to visualized project portfolio properties.

Figure 4.1 graphically illustrates the additional information available in the 2-dimensional visualization of project analogies compared to a traditional merely sorted list of similar projects. The list fails to indicate efficiently, how many similar projects should reasonably be considered, or whether a new project is highly isolated. This approach should give estimators an intuitive insight into portfolio data and exploit human cognition and pattern processing, thus achieving an accepted estimation method, as well as a better understanding of the correlation between data characteristics and estimation methods' accuracies.

- People can immediately assess the structure of portfolio data, especially the clusters of similar projects—this eases identification of outliers or unusual project behavior and allows for higher estimation accuracy and reliability. In addition, an estimate's confidence can easily be determined, for example, when the project to be estimated is similar to a large, dense cluster of projects performing similarly the estimator can be confident with an analogy-based estimate proposal.

- The method is visual, the mathematical model transparent, the process fast and easy-to-learn—this should guarantee high acceptance and low estimation effort. The interactive playing with the data set—i.e., choosing subsets of the data dimensions, zooming in on particular interesting project clusters—will enhance portfolio understanding and influence portfolio measurement.

More strictly, this thesis outlines expected benefits in the areas of model transparency, portfolio overview and understanding, selection of methodology, operational data handling, and estimation confidence assessment.

Section 2 refers to related work in the areas of cost estimation and MDS. Section 3 outlines the MDS approach and some quantitative criteria for assessing the approximation quality. Section 4 applies MDS to some real-world industrial project portfolio data sets. Section 5 discusses the potential benefits of the proposed visualization in the area of software cost estimation. Section 6 gives an outlook on further research directions in this field.

4.2 Estimators and Data Representation

Different approaches to software effort prediction have been proposed—algorithmic models like COCOMO 2 have been proposed by [BHM+00], neural networks are used by [Boe01], other methods rely on regression analysis (e.g., [SSS86]).

Several studies compare the different approaches' performance. [Kem87] reports potentially high error rates for COCOMO of up to 600 percent. [BLW00] compare different cost estimation techniques. The results illustrate the importance of defining appropriate similarity measures—without them the analogy method is outperformed by other methods. [WR02] have investigated the question whether multi-organizational data is of more value to software project cost estimation than company-specific data. Different methods like analogy, ordinary least squares (OLS) regression, and analysis of variance between groups (ANOVA) were used to predict costs for a large portfolio of multi-organizational project data. Results showed that if a company's project portfolio contains homogenous data, more accurate results can be achieved by analyzing the company's own data than by using large portfolios from external sources.

[SS97a] compare analogy-based approaches to regression analysis. Estimation results for regression methods and analogy are compared using a jack-knifing approach: one project is taken from the portfolio, its effort is predicted based on the remaining data, then the predicted effort is compared to the project's real effort; this is repeated for all projects. The result of this experiment was that analogy outperforms regression in most circumstances.

[MS99], however, come to a different conclusion. The authors design an environment where experienced and less experienced estimators have to estimate project effort using regression analysis or an analogy-based approach. A main result is that both regression and analogy can substantially improve an estimator's performance, but that regression analysis is not outperformed by analogy.

Several publications point out the importance of graphical representations in data mining environments [The01]. According to [RS02] the assimilation of unprepared, tabular information is not intuitive and visualization therefore aids the understanding and the extraction of features. According to [LS87] certain features are more easily extracted from diagrams than from tabular or sentential representations as diagrams can group together related concepts more easily than tabular representations. Tables may store related items in separate areas, which results in higher search effort to link concepts.

Joseph B. Kruskal, a psychometrist, was one of the first to work with MDS and authored many of the early publications [Kru64a, Kru64b, KW78]. [Lee01] offers a general introduction to MDS. Application fields for MDS, the different types of MDS, the different loss functions and algorithms are presented along with examples to illustrate the theoretical information. Another introductions to MDS is given [BG96].

MDS is used in a wide field of science disciplines. [CD82] present a collection with many of the classical MDS papers. [CC96] apply MDS methods to the field of information retrieval. [GSW95] use MDS for understanding brain connectivity.

Finally, early research results [AGB03a] indicate the feasibility of the proposed approach for several portfolio decisions and point out specific applications, especially cost estimation and portfolio standard compliance visualization.

4.3 Visualizing High-Dimensional Data

This section sums up the method of MDS and explains quantitative and graphical criteria for assessing its approximation quality.

MDS is a method to transform high-dimensional data to lower dimensions—usually in order to visualize it (e.g., with 2D-charts). MDS is based on the analogy or similarity of the visualized entities—in this case, software projects—, which are described as a vector of attributes or features. Originating from mathematical methods in psychology, MDS is gaining popularity in different areas such as medicine and knowledge management. We describe the procedure of preparing portfolio data, as well as an MDS tool in [AGB03a].

In particular, MDS offers several advantages over other multivariate statistical methods, as it (i) supports non-continuous, i.e., ordinal, data, (ii) allows for missing values, and (iii) makes no assumptions on the underlying data's distribution. These properties match typical properties of real-world data sets well.

The remaining section describes the following steps in applying MDS:

1. Prepare the portfolio data by selecting or weighting the data dimensions to cluster projects using the relevant dimensions.
2. Compute project dissimilarities to provide the input to the MDS visualization.
3. Visualize the dissimilarities using dedicated MDS tools.
4. Quantitatively assess the approximation quality of the MDS visualization and verify if the quality is within the boundaries of the MDS literature.

Sets of objects—in this case, projects—are characterized by the *dissimilarities*, i.e., distance-like quantities. The dissimilarities are denoted as δ_{ij} and are usually defined in a $n \times n$ *dissimilarity matrix*.

The importance of a dissimilarity δ_{ij} can be reflected by its *weight* w_{ij}. Distances in the lower-dimensional space $\mathbf{R}^m$ are denoted as $d_{ij}(X)$, with the *configuration* X representing the m coordinates of n entities in the m-dimensional space.

In order to compute the project dissimilarities, usually the Euclidean distance function is applied to two projects' features, where the feature values are first normalized to $[0-1]$, and $w_{ij} = 1$ (Note: in our case the features used to calculate the dissimilarity did not include the feature "effort"; this so-called *target feature* is depicted on the resulting MDS visualization). However, each feature or dimension would have the same impact on the dissimilarity, which is unlikely. One approach to weight the dimensions is to use a brute force approach to weight all dimension combinations and to assess each combination's mean magnitude of relative error (MMRE) value. A special case is weighting all combinations with 0 and 1, which is equivalent of selecting dimensions.

After selecting the dimensions' weights, the dissimilarity matrix can be computed using the Euclidean distance on the project dimensions, yielding the dissimilarity matrix. Then, tools are used to iteratively transform this matrix to coordinates in the lower-dimensional space $\mathbf{R}^m$.

In order to assess the approximation quality of an MDS visualization, a so-called stress value can be used. It compared the values of the original dissimilarities with the lower-dimensional distances to assess the degree, to which the new distances represent the original analogies or similarities in the high-dimensional feature space.

One example of a stress value function is *Kruskal's stress-1*; it gives the quality of the representation based on the square root of the squared errors of the representation compared with the disparities, divided by the sum of the squared distances on the representation:

$$\sigma_1 = \sqrt{\frac{\sum_{i<j} w_{ij}(\delta_{ij} - d_{ij})^2}{\sum_{i<j} w_{ij} d_{ij}^2}} \tag{4.1}$$

There is no general agreement on which value is acceptable; different authors define their own criteria. According to Kruskal's rule of thumb [Kru64a], a Kruskal stress-1 value of 0.2 reflects a poor fit between the distances and the dissimilarities, while a value of 0.1 is considered fair, 0.05 is good, 0.025 excellent and 0 is perfect.

A more detailed analysis is possible with Shepard diagrams; they visualize original project dissimilarities vs. distances in the two-dimensional graphical representation. Good approximations therefore produce almost linearly aligned data points.

4.4 Industrial Portfolio Data

In this section several high-dimensional real-world project portfolio data sets available in the public domain are visualized two-dimensionally using MDS. In addition, the approximation quality is assessed quantitatively and graphically. Please refer to the references given in table 4.1 for the original data sources.

Data sets could have been visualized using all the given dimensions; however, several dimensions contribute little or nothing to the clustering of projects. In the first step, the original number of dimensions was thus reduced by performing a brute-force search to achieve the optimal subset of dimensions. For this task we used the tool ArchANGEL[1] to select the subset of dimensions that minimizes the mean magnitude of relative error (MMRE) measure in a jack-knifing analysis. The MMRE value indicates how good an estimation approach is likely to perform in terms of accuracy or error percentage of estimated effort, in our case, ArchANGEL's analogy-based approach. However, this error value should rather be used to compare different approaches applied to the same data set, as it highly depends on the underlying portfolio data properties. Note, that the brute-force approach searches all combinations of dimensions by weighting them

[1]`http://dec.bmth.ac.uk/ESERG/ANGEL.`

Data set	Dimensions	Subset	MMRE
Albrecht [AG83]	5	4	0,635
Desharnais 1 [Des89]	9	1	0,368
Desharnais 2 [Des89]	9	3	0,388
Desharnais 3 [Des89]	9	3	0,343
Kemerer [Kem87]	2	1	0,676

Table 4.1: Visualized data sets [AGB04]

with either 0 or 1. A better result could be achieved by using a larger set of weight factors, for example (0, .25, 0.5, 0.75, 1).

In addition, dimensions describing project length or duration were excluded as these values are unlikely to be known at time of estimation.

Table 4.1 gives an overview of the data sets, giving the original number of data dimensions (including the feature "effort"), the optimal number of data dimensions according to ArchANGEL's procedure of searching all possible combinations of dimensions (excluding the feature "effort"), and the resulting MMRE value.

In the second step, the standard Euclidean distance function was applied to the normalized values of the selected features to calculate the dissimilarity matrix. This was performed by a custom spreadsheet macro.

In step 3, the dissimilarities were visualized using MDS (Note: only if the number of selected dimensions was greater than 2). In this thesis Addinsoft's Excel plug-in XLSTAT 6.1 and Miner3D were used.

Finally, table 4.2 lists the stress values of the data sets with more than two dimensions to be visualized. According to Kruskal's rule of

Data set	2D stress	3D stress
Albrecht	0.051	0.019
Desharnais 2	0.007	-
Desharnais 3	0.021	-

Table 4.2: Stress values [AGB04]

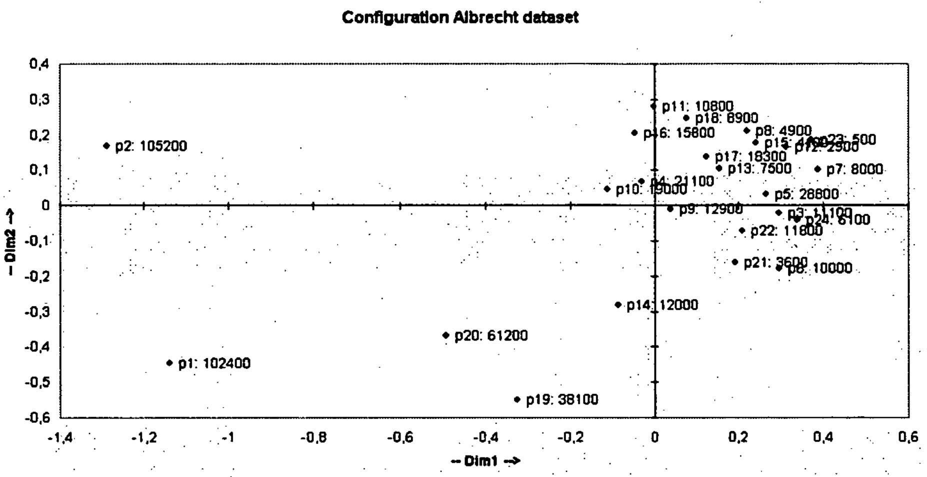

Figure 4.2: 2D MDS visualization of Albrecht data [AGB04]

thumb (see previous section), the given visualizations are between good and excellent with respect to the approximation to the original data; the Shepard diagrams support this impression.

Figure 4.2 depicts the MDS visualization of Albrecht's data set. As it can be seen, several projects (depicted in the left-hand part of the graph) are fairly different, thus distant, from the other projects. These projects (1, 2, 20) also have the highest effort values of the portfolio. The project arranged more densely on the graph's right-hand part are more similar to each other, but still contain several outliers with respect to their effort value, for example, project 5.

Figure 4.4 displays the Shepard diagram for this MDS visualiza-

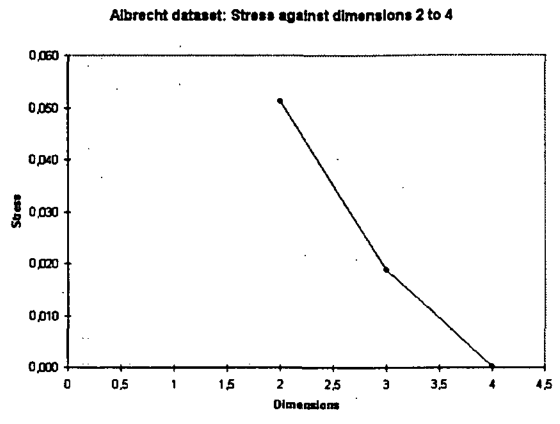

Figure 4.3: Albrecht stress

tion. It seems to support the impression of an good overall approximation quality.

Further figures (MDS visualizations of the Desharnais 2 and 3 data sets in tables 4.5 and 4.8; the respective Shepard diagrams in tables 4.7 and 4.10) are given in the appendix.

It is important to point out some limitations of the analogy-based approach and its visualization using MDS. First, the collected portfolio measurement data should be consistent. If collected by different persons using different procedures, data quality can be compromised; analysis relying on it has to fail. In our case, existing portfolio data sets were visualized, with little context information available about the data quality. Applying analogy-based approaches and MDS in an industrial environment would require careful data collection and verification procedures to ensure data quality.

Furthermore, some portfolios might not be suited for analogy-based analysis, especially if they comprise of mostly innovative projects, involving mainly new, unknown technology—the concept on analogy is simply not well-suited in environments dealing with singular projects.

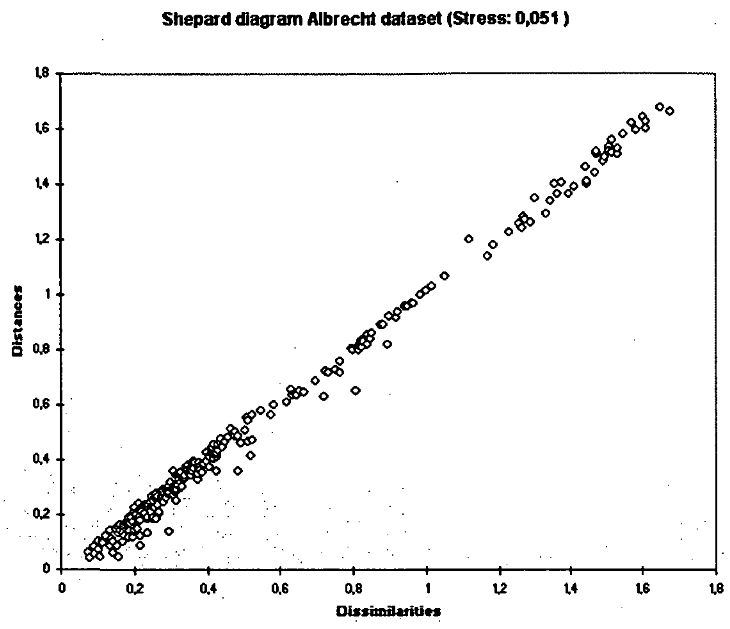

Figure 4.4: Shepard diagram of Albrecht data set [AGB04]

4.5 Discussion

Transparency is a key factor of estimation methods: without it, an estimate proposal's plausibility is unlikely to be assessable by human estimators—the model will remain unused.

This thesis proposed to enhance analogy-based approaches by visualizing high-dimensional portfolio measurement data with multi-dimensional scaling. In many circumstances, this is a feasible method to reproduce high-dimensional feature sets graphically; the approximation quality can be measured by the stress value. Data sets with 6 and more dimensions were visualized successfully within reasonable stress boundaries given in [Kru64a].

Previous attempt of applying MDS to portfolio data sets, like in [AGB03a], did not involve selecting just relevant project feature dimensions. In that case, the stress values obtained indicate that 3-

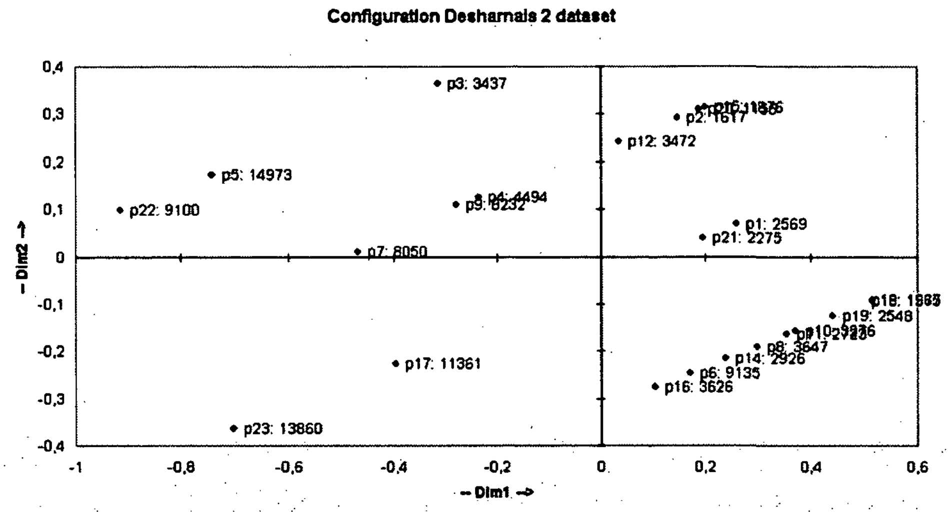

Figure 4.5: 2D MDS visualization of Desharnais 2 data [AGB04]

dimensional data representation might be necessary to obtain a suitable approximation quality. The approach presented in this chapter, i.e., selecting only a relevant subset of features, yield high approximation quality. Kruskal's stress values given in figure 4.2 are 2% to 5% for the 2-dimensional case, which can be considered—according to MDS literature—between good and excellent.

The following list should illustrate the qualitative benefits of the proposed approach with concrete examples visible in the portfolio visualizations:

- *Transparency.* The proposed method is straightforward and transparent; even estimators not acquainted with it immediately grasp the process and the visualizations' implications.

- *Overview.* MDS gives the user a visualization with a high information density. It is therefore easy to gain a fast overview of a project portfolio's properties.

  For example, while the projects in the Desharnais 2 data set

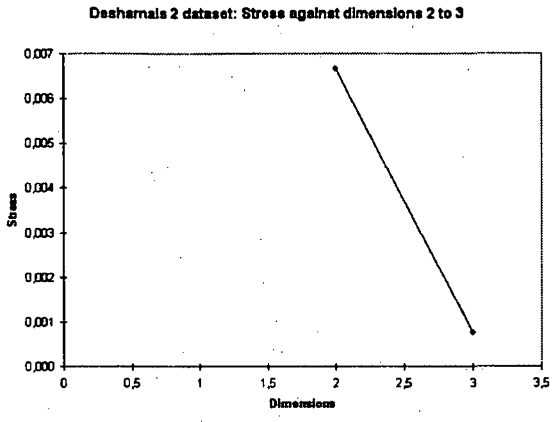

Figure 4.6: Desharnais 2 stress

form some clusters (see figure 4.5), the projects in the Desharnais 3 data set are less coupled.

- *Methodology.* Several publications comparing estimation methods indicate that no method can generally be regarded as the best one [Wie01]; a method's performance depends highly on the underlying portfolio data properties. Visualizing the data can help estimators to assess whether it is reasonable to apply analogy-based methods in a specific circumstance or whether a particular project cluster structure is unlikely to yield high-quality analogy-based estimates.

  For example, projects 6 and 10 in the Desharnais 3 data set (see figure 4.8) should probably not be estimated using the analogy-based approach as they are distant to the rest of the projects.

- *Operation.* The task of analyzing analogies in portfolio data involves identifying similar project feature sets. This can be performed fast and reliably on a visual representation of the data, especially as the criteria are varying (e.g., in some cases a larger cluster could be used as basis for the estimate if it is

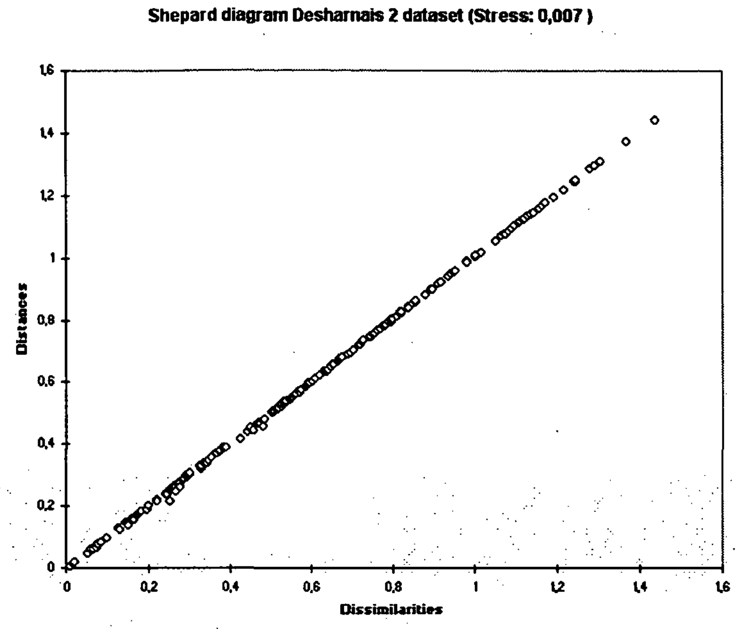

Figure 4.7: Shepard diagram of Desharnais 2 data set [AGB04]

dense, while in other cases instead of a fixed number of similar projects only one or few should be used due to a portfolio's high entropy).

For example, project 5 in the Albrecht data set (see figure 4.2) should probably not be allowed to influence estimates of nearby projects—its high effort value should first be analyzed to decide if this is a valid project to compare other projects to.

- *Confidence.* Finally, the benefits mentioned above (method transparency and user acceptance; coarse portfolio overview and understanding; assessment of a methodology's suitability; easy data selection and manipulation) contribute to increase the confidence in a particular estimation.

  For example, the lower right project cluster of the Desharnais

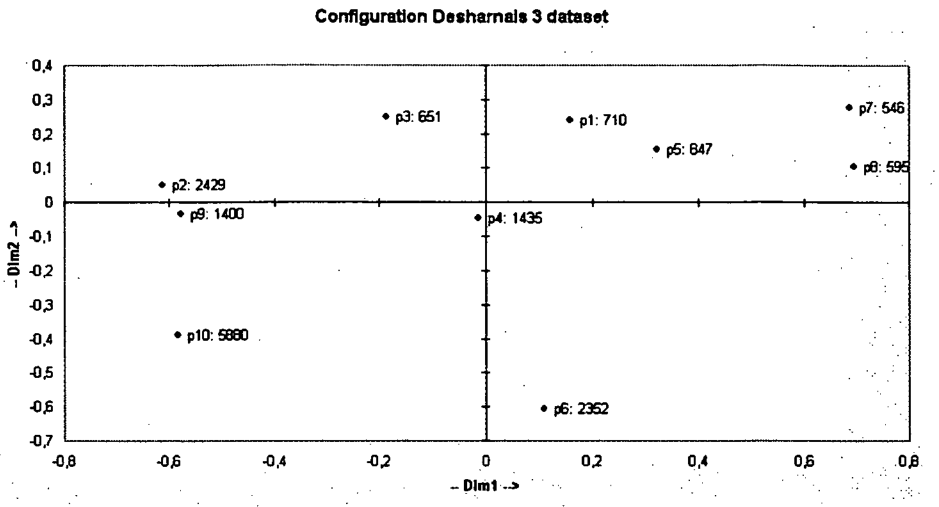

Figure 4.8: 2D MDS visualization of Desharnais 3 data [AGB04]

2 data set (see figure 4.5) seems—despite some outliers—to increase confidence in an effort estimate range between 2500 and 3500.

Chapter 7 outlines these qualitative benefits in greater detail.

4.6 Conclusion and Outlook

MDS provides a transparent method to visualize high-dimensional data and to analyze analogies or similarities intuitively. In this thesis we propose portfolio data preparation steps for an MDS visualization of high-dimensional project portfolio data, we visualize several real-world data sets and assess the achieved approximation quality, and we outline several benefits of the approach referring to concrete portfolio properties.

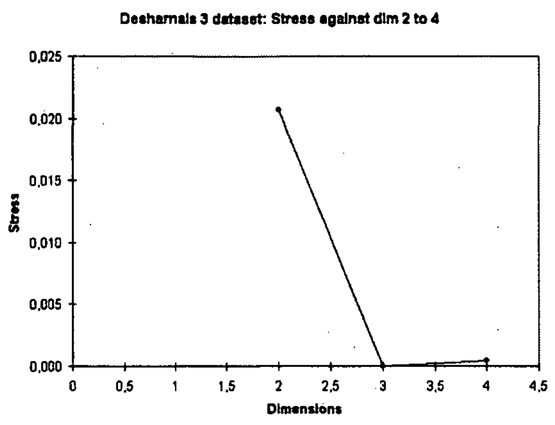

Figure 4.9: Desharnais 3 stress

Main findings are that the approximation quality is within reasonable boundaries given in the MDS literature, and that cost estimation can indeed benefit substantially from MDS—specific benefits include better transparency of the analogy-based approach, a better understanding of a portfolio's data properties, thus, easier assessment of the validity of analogy-based approaches in specific circumstances, easier data handling and project selection, and finally, higher confidence in estimates.

However, many aspects have to be refined and will be addressed in future research efforts. First, weighting portfolio data dimensions using brute force could be extended from the current approach to fine-grained weight levels. Second, user interface issues will be addressed to facilitate cluster analysis, for example, providing easy access to project cluster mean and variance values. Finally, quantitative measures for estimation confidence will be defined to assess the value of the visualization for the estimators, for instance, by weighting estimates' accuracies (post-project) with the estimators' corresponding confidence values in these estimates (pre-project).

To sum up, this and future research aims at supporting decision makers in the crucial task of cost estimation, by providing transparent

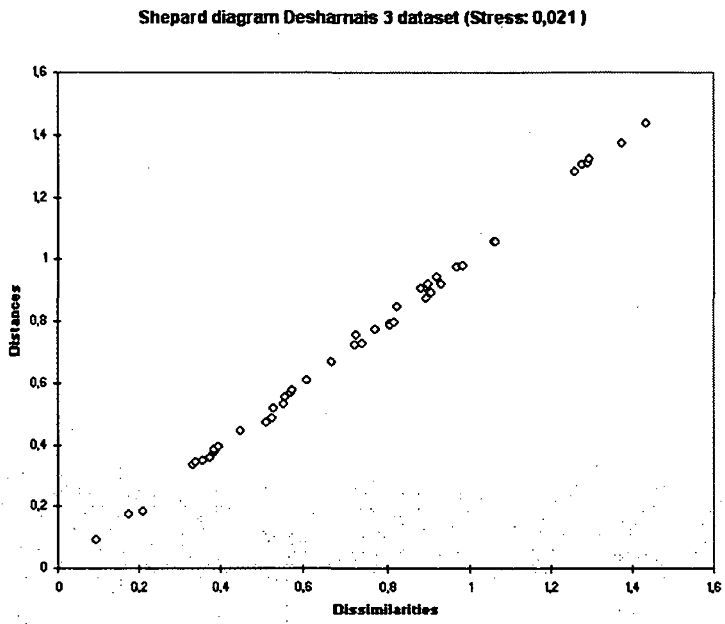

Figure 4.10: Shepard diagram of Desharnais 3 data set [AGB04]

and intuitive means to analyze portfolio data and assess estimates' plausibility.

Chapter 5

Estimation Effectiveness

Accurate and reliable software cost estimation is a vital task in software project portfolio decisions like resource scheduling or bidding. A prominent and transparent method of supporting estimators is analogy-based cost estimation, which is based on finding similar projects in historical portfolio data.

However, the various project feature dimensions used to determine project analogy represent project aspects differing widely in their relevance; they are known to have varying impact on the analogies—and in turn on the overall estimation accuracy and reliability—, which is not addressed by traditional approaches.

This chapter (a) proposes an improved analogy-based approach based on extensive dimension weighting, and (b) empirically evaluates the accuracy and reliability improvements in the context of five real-world portfolio data sets.

Main results are accuracy and reliability improvements for all analyzed portfolios and quality measures. Furthermore, the approach indicates a quality barrier for analogy-based estimation approaches using the same basic assumptions and quality measures.

5.1 Analogies and Features

Software cost or effort estimation [BHM$^+$00, Jon98] is a vital task in many prominent software project portfolio decisions, for example, in resource allocation, project start scheduling, bidding, and risk management. High-quality estimates are thus fundamental to a variety of success-critical project participants; typical quality criteria are estimation accuracy and reliability (i.e., the likelihood that an estimate is accurate).

A broad range of processes, procedures, mathematical methods and tools have been proposed to support cost estimation; many studies compare existing approaches—for example, regression models, expert judgment, or analogy-based methods—with respect to various estimation quality metrics, for example, the accuracy metric mean magnitude of relative error (a good overview is given in [Wie01]).

However, due to many constraints in empirical data collection—inconsistent quality of the collected software metrics, environments differing widely in domain or scope, different expertise level of estimators, software portfolio data properties due to small sample sizes, etc.—it remains difficult to generalize many of the obtained results, as pointed out by Myrtveit and Stensrud [MS99].

Nevertheless, several conclusions can be drawn safely:

- While formal estimation methods (like regression analysis or analogy-based approaches) are not the only relevant aspect of the estimation process, they do increase the estimators' performance [SM98].

- While there is no single best cost estimation method for all circumstances, analogy-based approaches rank among the best methods in a variety of environments, as pointed out by Wieczorek who gives an overview on previous study results [Wie01].

- Finally, while a method's estimation quality measures are important, other, secondary "usability" attributes—like transparency and simplicity—greatly affect a method's acceptance and applicability in practice; the analogy-based approach is both transparent and simple [SS97a, MC00].

To sum up: cost estimation is vital to software project portfolio management; cost estimation can be improved by supporting estimators with mathematical models and tools; cost estimation relying on the analogy-based approach is accepted and yields promising performance.

The traditional analogy-based approach is based on deriving estimates from historical project portfolio data by finding one or several projects (with known cost, effort, or productivity metrics) that are similar to the project to be estimated. Projects are regarded as "similar" if they feature similar project metrics or features (like number of project team members, expertise level, function point count, etc.). The degree of similarity is usually determined using the Euclidean distance on the l-dimensional feature space. However, traditional approaches do not address the features' different importance or weight adequately.

This thesis aims at extending analogy-based approaches as described by Shepperd and Schofield [SS97a], in order

1. to increase estimation accuracy as measured by the mean magnitude of relative error;
2. to increase estimation reliability as measured by the variance of the relative error;
3. to increase the model's stability as measured by the volatility of dimension weight values.

This is achieved by replacing conventional feature dimension *selection*—where subsets of features are determined for computing distances—by more flexible feature dimension *weighting*—where features are weighted differently and can thus influence the distance function with varying degrees—, and by implementing a tool for this more computation-intensive approach. Finally, we apply the approach to five industrial project portfolio data sets available in the public domain.

The resulting approach supports estimators better, as more accurate and reliable estimates are created, and as the model is more stable with respect to changes in the underlying software portfolio data, i.e., when adding new project data.

In addition, and maybe more importantly, if relying on the same basic assumptions (no outliers are removed from the portfolio; common estimation quality measures are used; the Euclidean distance is used; and merely the nearest projects are used to calculate the estimate), the new approach indicates an estimation quality barrier that can not be broken even by experienced estimators knowing about the different features' impact on the project similarities.

Section 2 refers to related work and gives a general overview on the cost estimation process. Section 3 describes the analogy-based approach and our extension, the research questions, and several estimation quality metrics indicating accuracy, reliability, and volatility. Section 4 presents the tool AMBER, developed to implement the proposed method, as well as the experimental setup. Section 5 gives an overview of the results when applying the new approach to several industry portfolio data sets. Section 6 discusses the research questions. Finally, section 7 concludes and suggests further research.

5.2 Potential and Pitfalls of Analogy

Cost and effort estimation is an iterative process involving several stages (for a process description see also Agarwal [AKY+01]). Figure

1.2 gives a simple process overview, describing how estimation is used in supporting decision evaluations in many project portfolio decisions like resource allocation or bidding. The estimation sub-process itself consists of the portfolio data collection, the estimate proposal creation with a model (and usually based on the collected data), and a calibration/sanity check by an estimator. Feedback mechanisms refine the data to be collected, or the collection procedure, as well as the estimate proposal creation step.

Many cost estimation research efforts focus on improving one specific aspect of the process—the creation of estimate proposals from past project data to human estimators. Different approaches to software cost prediction have been proposed—algorithmic models like COCOMO 2 [BHM+00], neural nets and regression trees [SF95], or checklists and group discussion [PS03].

Several studies compare the different approaches' performance using quantitative estimation quality measures [BLW00, FW97]. Kemerer [Kem87] reports potentially high error rates for COCOMO of up to 600 percent. A good overview on such comparative studies is given by Wieczorek [Wie01]. While some studies [MS99] analyze disclosed portfolio data sets, others [SS97a, Wie01] rely on data available in the public domain, like [AG83, Des89, Kem87], which makes it easier to compare or even generalize results. This thesis uses the latter approach.

Shepperd and Schofield [SS97a] compare analogy-based approaches to regression analysis. Estimation results for regression methods and analogy are compared using a jack-knifing approach: one project is taken from the portfolio, its effort is predicted based on the remaining data, then the predicted effort is compared to the project's real effort; this is repeated for all projects. The result of this experiment was that analogy outperforms regression in most circumstances.

Myrtveit and Stensrud [MS99], however, come to a different conclusion. The authors designed an environment where experienced and less experienced estimators have to estimate project effort using regression analysis or an analogy-based approach. A main result is that both regression and analogy can substantially improve an estimator's performance, but that regression analysis is not outperformed by analogy.

It appears as if a method's performance is highly sensitive to data quality and structure, estimators' experience level, experiment setup, etc. Nevertheless, estimation methods improve estimators' performance and thus the overall estimation process.

In addition to quantitative performance indicators, a method's usability (i.e., transparency) is fundamental as well for acceptance in practice [MC00]. Several more complex methods, for example, neural nets, fail to provide insight into how an estimate is produced [SS97a]. Analogy-based cost estimation is particularly well-suited to gain acceptance, as it is based on the same assumptions like conventional expert judgment—by far the most frequently used method [MJ03]. It simply relies on project analogy determined by the similarity of project features, which define a distance measure.

To sum up, estimation is a complex process with feedback mechanisms and a high degree of human interaction. A lot of research—including this thesis—is directed at improving one specific step of the process, the creation of estimate proposals from portfolio data. Further studies and carefully chosen experiment setting may measure the impact of highly optimized estimate creation methods on the overall estimation or even decision process—indeed this is necessary to fully understand and unleash improvements in estimation; this is not, however, within the scope of this thesis.

5.3 Analogy-Based Estimation and Dimension Weighting

This section describes the conventional analogy-based estimation approach in more detail, and its extension to address the different impacts of a project's features or metrics in determining similar projects in historical portfolio data; several quantitative measures are introduced or newly defined to allow for a comparison of the conventional and the new approach.

Analogy-based cost estimation [SS97a] is based—like basically all other estimation methods including neural networks—on one fundamental concept: *project efforts to be estimated will probably behave like efforts of similar past projects.* Thus, project features known at estimation time are used to look up similar past projects; the features to be estimated are derived from these past records.

5.3.1 Features and Estimates

Sticking to the notation used in [AGB03a, AB04], a project p—member of a project portfolio P—can be described by a tuple of features or attributes. Denoting features known at the project's estimation time with $d_1, ...d_l$, and the feature to be estimated (known after project completion) with e, a completed project is given by $P \ni p = \langle e, d_1, \ldots, d_l \rangle$.

To measure the degree of similarity or dissimilarity between two projects p and p', the Euclidean distance is used with modifications, i.e., the project feature dimensions are scaled to the interval $[0, 1]$, and they can be weighted with weights $w_i \in [0, 1]$:

$$\delta(p, p') = \sqrt{\sum_{i=1}^{l} w_i s_i^2 (d_i - d_i')^2} \tag{5.1}$$

$$s_i = \left(\max_{p \in P} d_i\right)^{-1} \tag{5.2}$$

By defining the mean effort e_Q of a portfolio $Q \subseteq P$ as $e_Q = |Q|^{-1} \sum_{p \in Q} e$, and by defining the set of most similar projects to p^* as $Q_{p^*} = \{p \in P | \delta(p, p^*) = min_{p \in P} \delta(p, p^*)\}$, a new project $p^* \notin P$ with known values $d_1^*, ..., d_l^*$ is given the following estimate:

$$e^* = e_{Q_{p^*}} \tag{5.3}$$

Current analogy-based approaches can be expressed with this or slightly modified models, for example, the strict equality relation in Q_{p^*}'s definition can be loosened to obtain a larger set of similar projects to calculate the mean effort from.

Thus, the estimate is calculated from past projects with minimal distance, usually a single project. If more than one project have the same minimal distance, their mean is used. The method can be extended to include even more past projects with small, yet not minimal distances.

5.3.2 Weighting Project Feature Dimensions

An important opportunity for improvement of current analogy-based estimation techniques is the treatment of the dimension weights w_i. These weight factors are usually either proposed to be set manually by experts (which involves difficult to formalize expert information impact), or—in the case of the tool ArchANGEL [SS97a]—are set to 0 or 1 in a brute-force search for the optimal selection of a dimension subset. The latter approach is a first approximation and will exclude irrelevant dimensions, but does not take into account the different importance or weight of relevant dimensions in determining project similarities.

Searching this optimal subset of dimensions is performed using a jack-knifing approach and estimation quality measures like the mean magnitude of relative error $MMRE$, or percentage of predictions that fall within 25% of the actual value ($Pred_{25}$).

For each $p^* \in P$, the estimate $e^* = e_{Q_{p^*} \setminus \{p^*\}}$ is calculated, yielding the relative error $r = (e^* - e)/e$ and

$$MMRE = |P|^{-1} \sum_{p \in P} |r| \tag{5.4}$$

$$Pred_{25} = |P|^{-1} |\{p \in P \big| |r| \leq .25\}| \tag{5.5}$$

One of these or another accuracy measure is calculated for all 2^l combinations of dimension selections; the best dimension combination is subsequently used to estimate new projects.

In this thesis, we propose to weight dimensions instead of merely selecting them, i.e., to use a range of possible dimension weight values w_i from $\{0, 1/(w-1), 2/(w-1), ..., (w-2)/(w-1), 1\}$, with w denoting the number of weights. For $w = 3$, the weights $\{0, 1/2, 1\}$ would be used. A simple algorithms can create all possible weight combinations to be tested.

The computational effort is larger, for the number of weight combinations is w^l. Still, it is possible to meet this extension's computational resource requirements for several reasons:

- The data sets usually contain only few projects, typically between 20 and 200; this allows for a fast computation of a single step.
- The number of dimensions is often low, too; the data sets used in this thesis have 3 to 7 dimensions.
- The weighting process is performed seldom, at completion of every project; it is not time critical.

- The computation can easily be parallelized.

5.3.3 Research Questions

The typical estimation quality criteria are accuracy and reliability (i.e., the likelihood of estimating accurately); improvements can be quantified using standard metrics, like those given in table 5.1. Analogy-based estimation performed in a real-world project portfolio should in addition analyze the impact of the different project feature dimensions, and its change at the addition of new project data.

We implemented a tool to perform the weighting algorithm and applied it to several industry portfolio data sets available in the public domain, comparing the conventional analogy-based approach (as given in [SS97a] to this thesis' extensive project feature dimension weighting approach). The results were used to address the following questions:

1.a Is the *MMRE* value *decreased* significantly? This would indicate a higher estimation accuracy.

1.b Using the *MMRE*-optimal dimension weights, is another accuracy indicator, namely, $Pred_{25}$ *increased*? This would further support a higher estimation accuracy, while a *decrease* would indicate that one accuracy measure—the *MMRE*—was optimized at the expense of another.

2. Is the variance of the relative error (denoted with Var_r) *decreased*? This would indicate a higher estimation reliability.

Two environments were used: (a) the five final project portfolio data sets, and (b) the data sets created by subsequently adding

Name	**Definition**
MMRE	Mean magnitude of relative error
$Pred_{25}$	Percentage of projects estimated within 25% of actual value
Var_r	Variance of relative error
MMDWC	Mean magnitude of dimension weight changes

Table 5.1: Estimation quality measures [AB04]

projects to small initial portfolios—thus analyzing the performance of the method over the typical life-time of a project portfolio data set.

In environment (b) a growing portfolio's optimal dimension weights w_i^t at the addition of new projects at stage t are further analyzed. When looking at the optimal dimensions obtained by the conventional selection approach, a single new project can completely change the selection of the dimensions, for example, from $w_1^8 = w_2^8 = 1, w_3^8 = 0$ to $w_1^9 = w_2^9 = 0, w_3^9 = 1$. In other cases, a dimension's weight oscillates between 0 and 1 at the addition of new projects to the portfolio, indicating that its real weight might be near 1/2. This unstable behavior is counterintuitive and makes estimate proposals more difficult to re-enact. By denoting portfolios obtained by adding new projects with $P_1 \subset P_2 \subset \ldots \subset P_n = P$, respective optimal weights w_i^t for a given P_t, and a suitable lower barrier t_0 defining the size of the initial portfolio, the simple volatility measure mean magnitude of dimension weight changes, $MMDWC = (n - t_0)^{-1} \sum_1^l \sum_{t=t_0}^{n-1} |w_i^t - w_i^{t+1}|$, can be defined to address the following question:

3. Is the volatility measure *MMDWC* decreased? This would indicate a more stable and less volatile model.

The answers to these questions are yes, yes with few exceptions, yes with few exceptions, and yes, respectively.

5.4 Extensive Feature Weighting Tool

In practice, dedicated tools have to be used to make analogy-based cost estimation feasible. The tool ArchANGEL [SS97a], for example, uses a brute-force approach to select the optimal subset of dimensions with respect to overall estimation quality measures like *MMRE*; in addition, it supports interactive dimension weighting.

As ArchANGEL's source code is not in the public domain, we programmed our dimension weighting tool, AMBER, from scratch, implementing the algorithms proposed in the previous section[1]. AMBER tries to emulate ArchANGEL in the trivial case of two possible dimension weight values, especially in the way how equidistant projects are treated. We compared AMBER's and ArchANGEL's results and noticed slight differences in some isolated portfolio settings, probably due to variations in the floating-point arithmetic; as these variation were negligible we used AMBER to reproduce the conventional approach in our test settings (as it has a better user interface for batch mode operation).

AMBER is a Java command line tool to facilitate batch processing of different portfolio and possible weight value combinations. AMBER's input is a data file with tab separated columns: project number, effort e, and arbitrarily many dimensions d_i (this format is compatible with ArchANGEL). AMBER produces a single-line output: the optimal *MMRE* value; the Pred_{25} value of the *MMRE*-optimal dimension weighting; Var_r, i.e., variance of relative error, of the *MMRE*-optimal dimension weighting; the dimension weight values w_i yielding the optimal *MMRE* value; and—for easier data processing in OLAP or spreadsheet applications—several variables like number of detected projects/dimensions, name of input file, and number of weight values used. Furthermore, AMBER allows a subset of projects to be excluded from analysis to ease stability analysis and the exclusion of outliers.

AMBER's brute force approach is able to handle the relatively

[1]Please contact the authors to obtain AMBER's source code.

small portfolio data sets, even though implemented in Java and not being fully optimized. The test's largest data set, Desharnais 1 (see table 5.2), featuring 44 projects and seven dimensions, was analyzed using 9 possible dimension weight values—yielding approximately 5 million dimension weight combinations—in about 30 minutes on a 1GHz single processor system.

Five real-world project portfolio data sets available in the public domain—refer to [AG83, Des89, Kem87]—were analyzed using AMBER. Table 5.2 gives an overview on the portfolio data properties: the number of projects, and the number of dimensions or features known at estimation time, i.e., excluding effort or productivity dimensions. The Desharnais data sets were obtained by splitting up the original data sets which contained projects from 3 different development environments.

The data sets contain merely quantitative metrics (function points, team experience, etc.) known or estimable at project start time. Project duration metrics were treated as not being available at estimation time; they were excluded from the estimation procedure (the number of dimensions in table 5.2 does not include such metrics, either).

5.5 Empirical Evaluation

This section presents the results obtained by applying the tool AMBER —implementing the extensive dimension weighting—to five industrial portfolio data sets. The application to the complete portfolio data sets yields the estimation quality metrics *MMRE*, $Pred_{25}$, and Var_r. In a second setting, growing portfolios are analyzed, yielding the same metrics, and, in addition, the volatility measure *MMDWC*, which describes the volatility of the dimension weights in growing portfolios. The *MMRE* and $Pred_{25}$ values indicate changes in estimation accuracy; Var_r quantifies changes in estimation reliability;

Data set	**Dimensions**	**Projects**
Albrecht [AG83]	6	24
Desharnais 1 [Des89]	7	44
Desharnais 2 [Des89]	7	23
Desharnais 3 [Des89]	7	10
Kemerer [Kem87]	3	15

Table 5.2: Public domain portfolio data sets [AB04]

finally, *MMDWC* is used to analyze the changes in optimal feature dimension weights in a typical growing portfolio environment. All measures are defined in section 3.

The data should answer, *if* several quantitative estimation quality indicators are affected by the extensive dimension weighting; if yes, *to what extent*, indicating a barrier for analogy-based estimation relying on the conventional assumptions of distance and measures; and whether more flexible dimension weights would reduce the high dimension weight volatility of the conventional approach.

Table 5.3 gives an overview on the accuracy and reliability metrics *MMRE*, Pred_{25}, and Var_r of the conventional approach using 2 possible weight values (0 and 1) compared to the new approach with 9 possible weight values (0, 1/8,...,7/8, 1), in 5 different real-world portfolio data sets. Note that positive percentage values denote *improvement* of a metric, i.e., *MMRE* reduction, Pred_{25} increase, and Var_r reduction, respectively.

Note: adding even more weight values would have substantially increased the computational effort in our case—as we tested many portfolios and portfolio-subsets—,while not yielding significant further improvements.

This snapshot describes the performance of the improved technique on the complete portfolio data sets (for portfolio size information, please refer to table 5.2); a more detailed analysis is possible if the metrics are computed on the growing portfolio, where portfolios

	Alb.	**Des. 1**	**Des. 2**	**Des. 3**	**Kem.**
MMRE 2 weights	0.602	0.368	0.361	0.347	0.676
MMRE 9 weights	0.482	0.334	0.306	0.263	0.582
MMRE reduction	20%	9%	15%	24%	14%
$Pred_{25}$ 2 weights	0.375	0.477	0.522	0.500	0.267
$Pred_{25}$ 9 weights	0.500	0.477	0.522	0.500	0.334
$Pred_{25}$ increase	33%	0%	0%	0%	25%
Var_r 2 weights	1.699	0.246	0.229	0.168	0.767
Var_r 9 weights	1.043	0.187	0.134	0.098	0.597
Var_r reduction	39%	24%	41%	42%	22%

Table 5.3: Accuracy and reliability metrics in five industrial portfolios using 2 vs. 9 possible dimension weight values [AB04]

	Alb.	**Des. 1**	**Des. 2**	**Des. 3**	**Kem.**
Md. *MMRE* reduction	16%	12%	12%	22%	0%
Md. $Pred_{25}$ increase	10%	10%	20%	27%	0%
Md. Var_r reduction	22%	24%	16%	32%	0%

Table 5.4: Median improvements of accuracy and reliability metrics in growing portfolios [AB04]

are built by adding one project after the other to an initial portfolio. This yields more portfolio data sets, and models the way portfolio data is used in the real-world as a growing historical database. Table 5.4 lists median changes in the accuracy and reliability metrics, measured in each intermediate portfolio of a growing portfolio. Again, positive values denote an improving metric.

Note: the projects' numerical identifiers were used to establish the chronological order of the procedure of adding projects to the portfolio. This is the exact procedure which would arise in a real-world environment where new projects are first estimated in light of the past portfolio data, and then, after completion, they are added to contribute to the knowledge base.

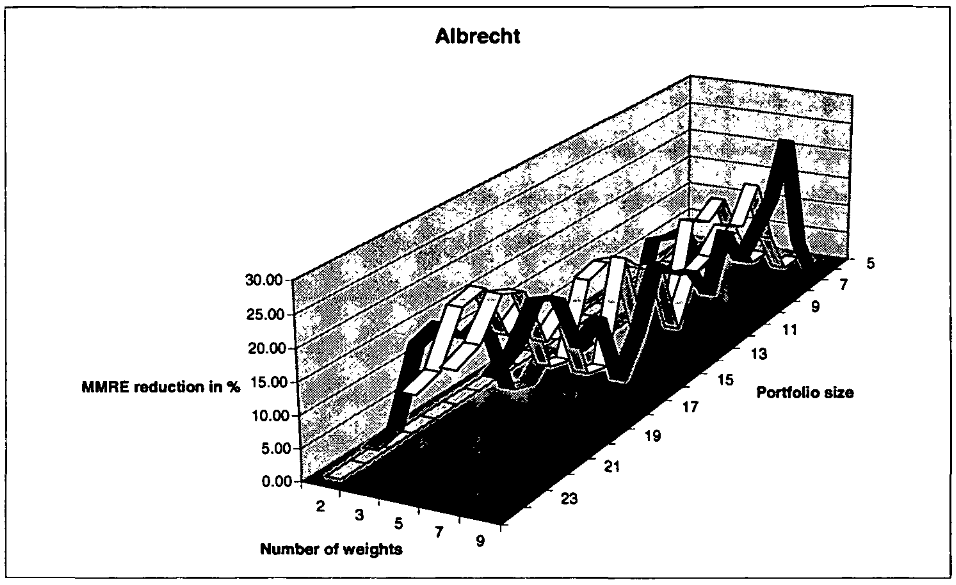

Figure 5.1: *MMRE* reduction with Albrecht data set [AB04]

Figures 5.1 and 5.2 show, how different numbers of weights affect improvement of the *MMRE* value in a growing portfolio. Improvement differs due to the underlying portfolio's data properties; however, the improvement is substantial across the complete range of both growing portfolios.

Giving another view, the box plots in figures 5.3, 5.4 and 5.5 display the obtained *MMRE*, $Pred_{25}$ and Var_r *improvement* in % for the Desharnais 3 data set, and for 3, 5, 7 and 9 possible dimension weight values. Generally, the measures improve as more possible dimension weights are added, indicating higher estimation accuracy and reliability.

The initial portfolio size was set to 5; smaller portfolios tend produce arbitrary results and are not suited for analogy-based analysis.

Table 5.5 shows the volatility measure *MMDWC*, which describes the average magnitude of weight changes when adding new projects to portfolios. Figures 5.6 and 5.7 depict the absolute weight values—and

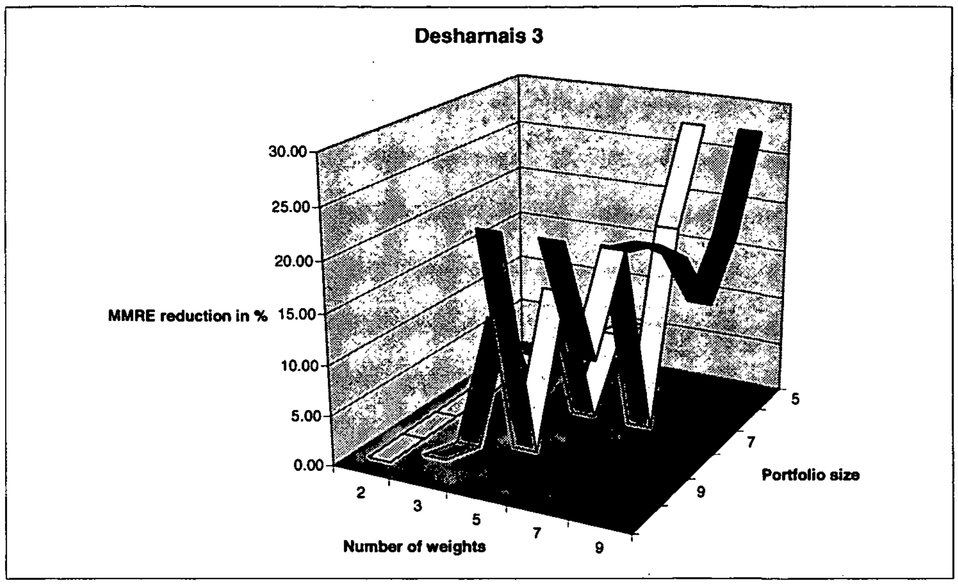

Figure 5.2: *MMRE* reduction with Desharnais 3 data set [AB04]

thus their changes—in more detail for the Desharnais 2 data set.

	Alb.	Des. 1	Des. 2	Des. 3	Kem.
MMDWC 2 weights	1.42	1.79	1.72	2.80	0.40
MMDWC 9 weights	0.93	1.03	0.60	2.30	0.26
MMDWC reduction	35%	42%	65%	18%	35%

Table 5.5: Volatility measure *MMDWC* reduction using 2 vs. 9 possible dimension weight values [AB04]

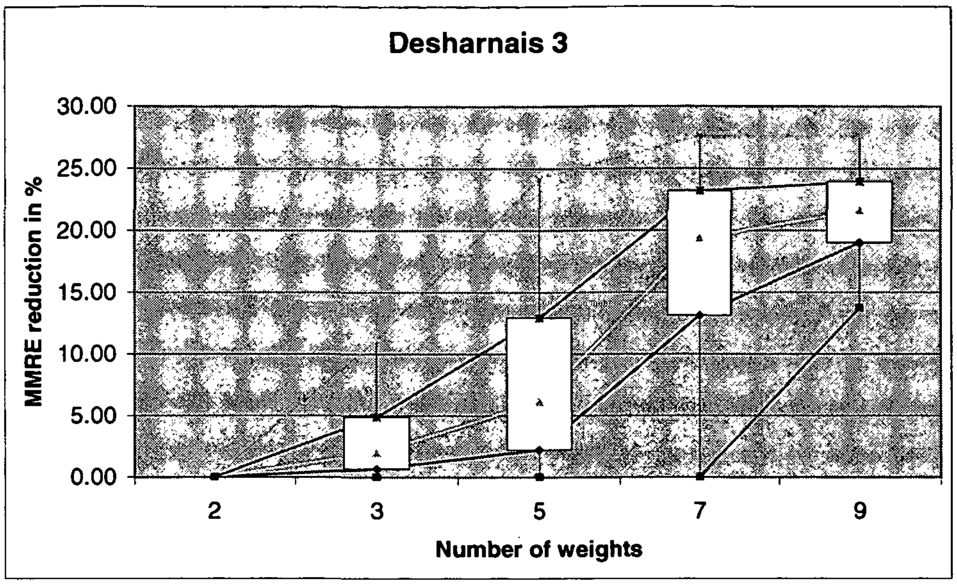

Figure 5.3: *MMRE* reduction with Desharnais 3 data set [AB04]

5.6 Discussion

The following questions concerning estimation accuracy, reliability and volatility were raised:

1.a Is the *MMRE* value *decreased* significantly, indicating higher estimation accuracy?

1.b Using the *MMRE*-optimal dimension weights, is another accuracy indicator, namely, $Pred_{25}$ *increased*, thus further supporting a higher estimation accuracy?

2. Is the variance of the relative error (denoted with Var_r) *decreased*, indicating higher estimation reliability?

3. Is the volatility measure *MMDWC* decreased, indicating a more stable model?

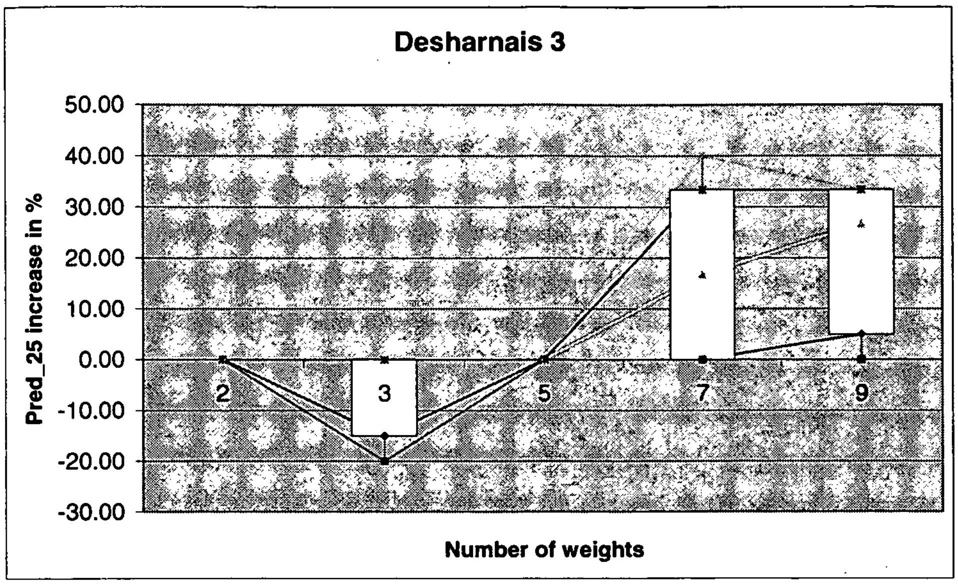

Figure 5.4: *$Pred_{25}$* increase with Desharnais 3 data set [AB04]

Accuracy. Using more than 2 dimension weight values, the *MMRE* value was decreased for all 5 portfolio data sets, up to 24%. Applying the new approach to growing portfolios, i.e., portfolios created by adding new projects to an initial portfolio, further supports this impression—the median *MMRE* reduction was up to 22% for the Desharnais 3 data set.

Note, that the initial portfolio size $t_0 = 5$ was chosen; using larger initial portfolios would further benefit the median *MMRE* reduction, as the new approach performs similarly to the conventional at the initial stages, when the portfolio is small.

In the case of the Kemerer data set, median values of the estimation quality measures were not affected (see table 5.4); this is due to the fact that only when most projects of this set are used, the new approach improves the estimation quality (14%, for example, in the case of the total portfolio, see table 5.3).

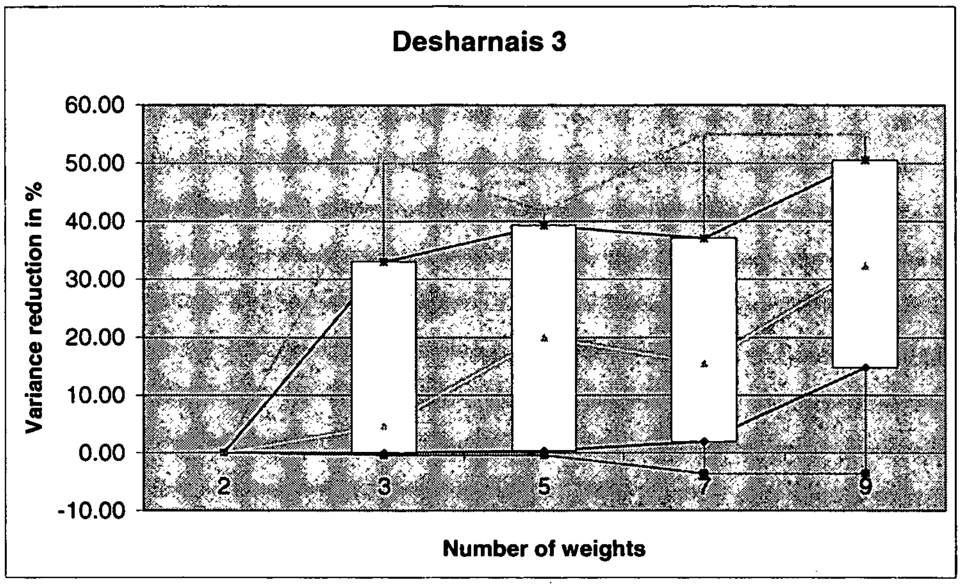

Figure 5.5: Var_r reduction with Desharnais 3 data set [AB04]

Another commonly used measure for estimation accuracy is *Pred*$_{25}$. Using the *MMRE*-optimal dimension weights, a portfolio's *Pred*$_{25}$ value was calculated to find out whether it would support the impression of higher estimation accuracy indicated by the primary measure, *MMRE*.

At first, the values given in table 5.3 do not seem to indicate increased *Pred*$_{25}$ values, as only 2 of 5 portfolios feature an increase. This picture, however, is distorted—when analyzing the *Pred*$_{25}$ values in detail (see, for example, figure 5.4). 4 of 5 data sets have a median *Pred*$_{25}$ increase between 10% and 27% (see table 5.4). In our environment, two cases were observed, when *MMRE*-optimized dimension weight negatively affected the *Pred*$_{25}$ measure, for example, in the case of 3 dimension weight values and the Desharnais 3 data set (see figure 5.4). However, increasing the number of dimension weight values to 7 or 9 yields increased *Pred*$_{25}$ values for all portfolios.

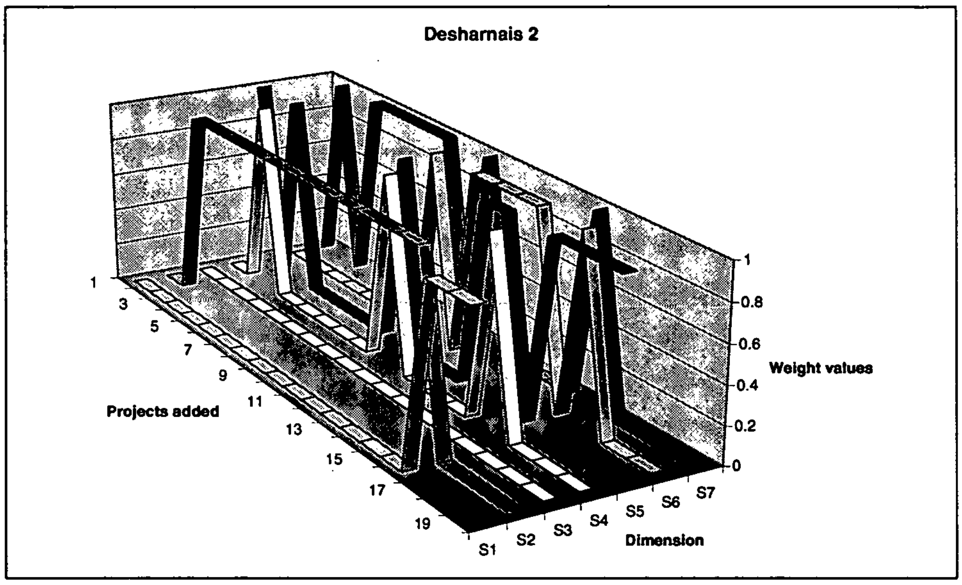

Figure 5.6: Dimension weight values of Desharnais 2 data set using 2 weights [AB04]

Thus, $MMRE$-optimal dimension weights usually have better $Pred_{25}$ performance measures, too. Both measures support the impression of improved estimation accuracy.

Reliability. The variance of the relative error Var_r can be used as a measure to indicate an estimation method's reliability. As it can be seen in table 5.3, the variance is reduced up to 42% in the case of the Desharnais 3 data set; the median value in a growing portfolio up to 32%, see table 5.4. Figure 5.5 illustrates, that the more dimension weight values are used, the more the Var_r value is improved, i.e., reduced.

Volatility. As pointed out in section 2, conventional dimension weighting (i.e., selection) is very sensitive to outliers, which frequently results in oscillating weight values when adding new projects to port-

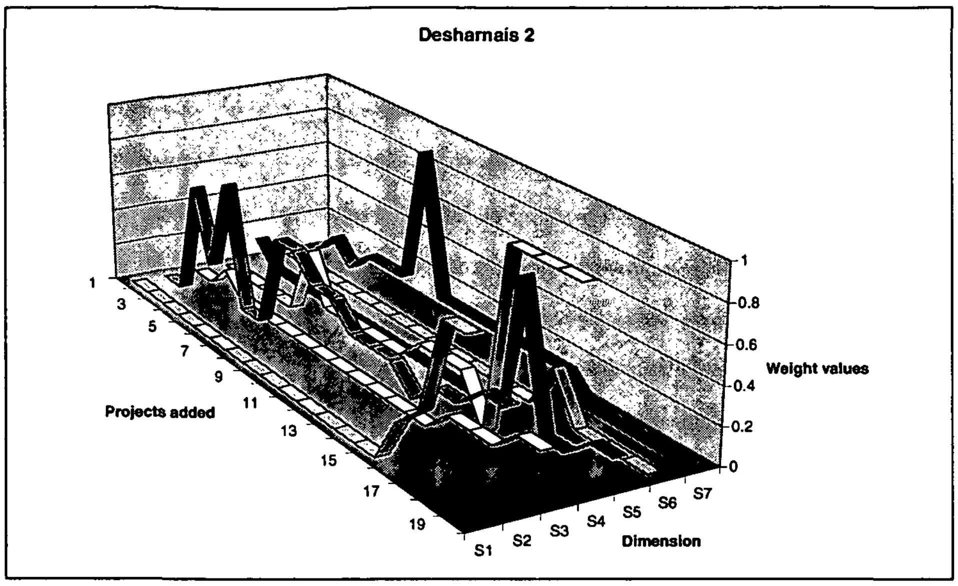

Figure 5.7: Dimension weight values of Desharnais 2 data set using 9 weights [AB04]

folios. Figure 5.6 illustrates how weight values change when new projects are added to the Desharnais 2 data set until the set is complete.

When relying on more dimension weight values, no mere binary on/off switching of dimension weights takes place; figure 5.7 illustrates the weight changes of the same Desharnais 2 data set, using 9 dimension weight values. Despite some peaks it is clearly visible that dimension weights are oscillating less.

In order to quantify this volatility, a simple measure—as used in time series analysis—was introduced in section 2, the mean magnitude of dimension weight changes *MMDWC*. For all data sets, this measure is reduced substantially, up to 65%.

Thus, the questions can be answered: yes, *MMRE* is reduced, $Pred_{25}$ is increased, both indicating a higher estimation accuracy;

yes, Var_r is reduced, indicating higher estimation reliability; yes, $MMDWC$ is reduced, indicating a more stable and less volatile method.

Chapter 7 offers a more detailed interpretation of these results.

5.7 Conclusion and Outlook

Analogy-based cost estimation is a valuable support for estimators; it is based on the concept of finding analogous or similar projects in historical portfolio data sets.

However, the project feature dimensions to derive similarity measures between projects are known to have varying impact on actual project similarity. In this thesis we address this issue by extensively searching optimal dimension weight values with respect to common estimation quality measures. For evaluation of the changes we tested the improved technique on five real-world portfolio data sets.

Main conclusions are: the new approach increases estimation accuracy and reliability regarding the measures $MMRE$, $Pred_{25}$ and Var_r; furthermore, the volatility of the conventional project feature dimension selection is reduced as weighting dimensions offers more degrees of freedom than the conventional binary dimension selection approach.

In addition to these improvements, the results indicate a barrier of estimation quality in analogy-based estimate proposal creation: using the same basic assumptions on quality measures, distance functions, etc., not even experienced estimators knowing about the impact of the different project features can improve the selection of data from historical project portfolio data sets.

Further research will focus on outlier analysis, and on visualization methods to increase human assessment of an estimate's plausibility.

Chapter 6

Estimation Efficiency

Software cost estimation is a ubiquitous task in software project portfolio management. Analogy-based estimation approaches are particularly suitable to reuse past project experience to create estimate proposals.

However, to find project data of past, similar projects, one usually must rely on similarity measures based on explicit project metrics, like a team's project experience, the number of interfaces, etc. This requires the consistent collection of such project metrics over time, which can be tedious, error-prone and expensive.

This thesis proposes an alternative approach: to use text artifacts arising in the regular course of a project to determine the similarity of project entities. Use cases' text descriptions are used as entities; a variation of self-organizing maps is used to determine their similarities.

The thesis points out possible applications and limitations of the approach, analyzes several real-world use case sets, describes problems and suggests rules for writing suitable use cases.

6.1 Minimizing Measurement Needs

Most relevant software portfolio decisions—like resource allocation, bidding, or start scheduling—rely on supporting procedures like cost estimation or risk assessment. Usually, a key for successful decision support is seen in software and process measurement, i.e., the definition, collection, storage and analysis of past project features, preferably in quantitative metrics.

Yet in many industrial environments such explicit measurement programs are never implemented, or fail. The reasons, outlined in more detail in the following section, are manifold, but they can be summed up: project team members often perceive such programs as economically inefficient, at least in the short term.

However, in certain circumstances *explicit* measurement might be unnecessary; instead, *implicit* metrics can be used, merely derived from artifacts that are created anyway during the development process, yielding valuable information without or with negligible extra cost.

One area we deem suitable for such an approach is analogy-based cost estimation. It tries to identify past projects that are similar to the project to estimate, and to create an estimate proposal by using the historical cost data. Traditionally, analogies are defined as distances of explicitly collected project features or metrics.

This thesis proposes to use the textual description of requirement documents—more precisely, structured chunks of documents, use cases—to determine implicitly the similarity of various requirements, allowing for a fast search of similar past project requirements without explicit project metrics [ABRB05]. We try to evaluate this approach on several industrial use case sets, and find out, whether those sets can be used without substantial modification in writing them.

Section 2 describes related work on cost estimation, use cases and digital libraries, with section 3 introducing self-organizing maps, used to organize the use cases by semantic similarity. Section 4 presents the use cases sets used. Section 5 gives the results. Section 6 discusses the lessons learned and outlines the benefits of the approach. Finally, section 7 gives an outlook on further research in this area.

6.2 Applying Digital Library Concepts to Use Cases

Cost estimation [Jon98] is a vital task in software project portfolio management. A variety of different methods have been proposed to support cost estimation: algorithmic models like COCOMO 2 [BHM+00], neural networks [Bod98], expert judgment [Hug96], or analogy-based approaches [SS97a].

The fundamental principle of those methods is that similar projects are likely to have similar cost characteristics—this is expressed most directly by the analogy-based approach, where estimates are derived from historical project information. One or several similar past projects are selected according to a simple distance function δ on the l project features or metrics $d_1, \ldots, d_l$ (scaled to a unit interval with s_i, and weighted according to a metric's influence with w_i):

$$\delta(p, p') = \sqrt{\sum_{i=1}^{l} w_i s_i^2 (d_i - d_i')^2} \tag{6.1}$$

The actual costs of the selected projects with the smallest distance are then used to estimate the new project; visualization methods like multidimensional scaling [AGB04] can be used to further support estimators.

While this method is straightforward and highly transparent (considered an important feature for the real-world application of a cost estimation method [FW97]), it has a main drawback: it requires the consistent collection of software project metrics over a long period of time, as only company-specific data can reasonably be expected to yield high-quality estimates [JRW00].

However, defining and collecting such software metrics explicitly is notoriously difficult. Maxwell [Max01] describes several common pitfalls in software measurement. Jeffery et al. [JRW00] describe data collection as an "expensive and time-consuming process for individual organizations." Auer et al. [AGB03b] point out, that the necessary automation of data collection can be difficult to achieve in highly heterogeneous software development environments. Ruhe et al. [RJW03] declare that as data collection has to be carefully planned, many organizations simply "do not have enough resources for the required measurement methods."

It would be preferable to assess project or project artifact similarities implicitly, without relying on explicit metrics. One possibility is to measure textual similarities between text documents created in the development process. Users benefit because similar documents are uncovered in an intuitive way, with no need for additional metric data collection. This approach is evaluated within the framework of the *SOMLib* project[1] [RM99, RM03]. The *SOMLib* project is based on unsupervised neural network technology for the task of document archive organization. In particular, the approach relies on the *Self-Organizing Map* (SOM) [Koh82, Koh95, KKL$^+$00], and variants thereof, such as the GHSOM [RMD02], providing a map-based representation of a document archive. In such a representation, documents dealing with similar topics are located next to each other. The obvious benefit for the user is that navigation in the document archive is similar to the well-known task of navigating in a geographical map.

Certainly, it is not suitable to compare whole project requirement documents to each other—yet standardized, smaller text fragments could be used, if the way they are created is sufficiently standardized. A potential candidate are *use cases* [Coc00], which are gaining importance in eliciting and documenting user requirements. The use case

[1]The *SOMLib* project homepage is available at `http://www.ifs.tuwien.ac.at/~andi/somlib`

methodology is applied in a variety of areas, ranging from e-commerce applications [SJP02] to embedded systems [NMB02]. Other potential text artifacts suitable for this approach include user stories or scenarios; however, this thesis analyzes industrial use case sets.

6.3 Text-Based Analogies

Self-organizing maps (SOM) are a way to map high-dimensional feature sets to lower-dimensional representations. This section describes the SOM approach in detail.

6.3.1 Indexing

In order to utilize the SOM for organizing documents by their topic, a vector-based description of the content of the documents is created. Instead of manually or semi-automatically extracted content descriptors, a rather simple word frequency based description is sufficient to provide the necessary information. For this word frequency based representation a vector structure is created consisting of all words appearing in the document collection. This list of words is usually cleaned from so-called stop words, i.e., words that do not contribute to content representation and topic discrimination between documents. Simple statistics allow the removal of most stop words in a very convenient language- and subject-independent way (words appearing in too many documents can be removed without the risk of loosing content information; words appearing in less than a minimum number of documents can be omitted, as well).

The documents are described within the resulting feature space of commonly between 2,000 and 15,000 dimensions, i.e., distinct terms they are made up of. Binary indexing may be used to describe the content of a document by simply stating whether a word appears in the document, or more sophisticated schemes can be used, such as *tf* × *idf*, i.e., term frequency times inverse document frequency [Sal89]. This weighting scheme assigns higher values to terms that appear frequently within a document, i.e., have a high term frequency, yet rarely within the complete collection, i.e., have a low document frequency. Usually, the document vectors are normalized to unit length to make up for length differences of the various documents.

6.3.2 Self-Organizing Maps

The self-organizing map [Koh95] provides cluster analysis by producing a mapping of high-dimensional input data $x \in \mathbf{R}^n$, onto a usually 2-dimensional output space while preserving the topological relationships between the input data items as faithfully as possible. This model consists of a set of units, which are arranged in some topology, where the most common choice is a two-dimensional grid. Each of the units i is assigned a weight vector m_i of the same dimension as the input data, $m_i \in \mathbf{R}^n$. These weight vectors are initialized with random values.

During each learning step, the unit c with the highest activity level, i.e., the *winner* c with respect to a randomly selected input pattern x, is adapted in a way that it will exhibit an even higher activity level at future presentations of that specific input pattern. Commonly, the activity level of a unit is based on the Euclidean distance between the input pattern and that unit's weight vector. The unit showing the lowest Euclidean distance between it's weight vector and the presented input vector is selected as the winner.

Adaptation takes place at each learning iteration and is performed as a gradual reduction of the difference between the respective components of the input vector and the weight vector. The amount of adaptation is guided by a learning rate α that is gradually decreasing in the course of time. This decreasing nature of adaptation strength ensures large adaptation steps in the beginning of the learning process where the weight vectors have to be tuned from their random initialization towards the actual requirements of the input space. The ever smaller adaptation steps towards the end of the learning process enable a fine-tuned input space representation.

As an extension to standard competitive learning, units in a time-varying and gradually decreasing neighborhood around the winner are adapted too. The neighborhood of units around the winner may be described implicitly by means of a (Gaussian) neighborhood-kernel h_{ci} taking into account the distance—in terms of the output space—between unit i under consideration and unit c, the winner of the current learning iteration. This neighborhood-kernel assigns scalars in the range of [0, 1] that are used to determine the amount of adaptation ensuring that nearby units are adapted more strongly than units farther away from the winner. A Gaussian may be used to define the neighborhood-kernel, where $||r_c - r_i||$ denotes the distance between units c and i within the output space, with r_i representing the two-dimensional vector pointing to the location of unit i within the grid:

$$h_{ci}(t) = e^{-\frac{||r_c - r_i||^2}{2 \cdot \delta(t)^2}} \tag{6.2}$$

It is common practice that in the beginning of the learning process the neighborhood-kernel is selected large enough to cover a wide area of the output space. The spatial width of the neighborhood-kernel is reduced gradually during the learning process such that towards the end of the learning process just the winner itself is adapted. Such a reduction is done by means of the time-varying parameter δ. This strategy enables the formation of large clusters in the beginning and fine-grained input discrimination towards the end of the learning process.

With α representing the time-varying learning rate, h_{ci} representing the time-varying neighborhood-kernel, x representing the currently presented input pattern, and m_i denoting the weight vector assigned to unit i, one obtains the following learning rule:

$$m_i(t+1) = m_i(t) + \alpha(t)h_{ci}(t)[x(t) - m_i(t)] \tag{6.3}$$

Pragmatically speaking, during the learning steps of the self-organizing map a set of units around the winner is tuned towards the currently presented input pattern enabling a spatial arrangement of the input patterns such that alike inputs are mapped onto regions close to each other in the grid of output units. Thus, the training process of the self-organizing map results in a topological ordering of the input patterns.

An extension to SOM, especially for large, growing sets of documents, is the growing hierarchical SOM. Additional parameters determine the behavior of the grow process. Please refer to [RMD02] for more details.

6.4 Use Case Set Description

Unsurprisingly, it is non-trivial to get access to real-world instances of use cases in industrial environments. While several companies and institutions do actively apply use cases in describing user requirements (indeed, more than we initially expected), they are reluctant to give access to those use case sets, often even under terms of non-disclosure agreements. Similar experiences were indicated to us by several academics working in the field of use case applications.

In addition, some basic criteria had to be met by the use case sets, further narrowing the number of settings for analysis:

- *Size.* A set of use cases should contain sufficient use cases to make analogy-based estimation feasible. Several sets containing less than 20 use cases were dismissed.

- *Relevance.* A set should consist of real-world use cases, used in industrial environments, and not be artificially constructed. It should involve large institutions with a software engineering track record, not just small software producers.

- *Structure.* The use cases should more or less conform to structured use case templates and not be totally informal text fragments describing some ill-defined system aspects.

- *Readiness.* The use cases should—except for conversion to txt-files, and replacement of special characters—be ready to use with the neural network techniques applied; it should not be necessary to prepare or substantially change the use cases—this would not be possible in a real-world environments and reduce the efficiency gains of implicit measurement of existing artifacts to nothing.

We finally obtained access to four suitable sets of use cases; they are described in the following. Company names and details had to be omitted due to the terms of confidentiality agreements.

- *Set 1: TICKET.* TicketLine is a ticket reservation system developed at the Vienna University of Technology, primarily by graduate students, for use in both graduate and undergraduate courses. Three main releases were developed over the years, each with state-of-the-art methods and terminology; release 3 used use cases as primary requirement description approach. An earlier version can be found on CD in [ZBGK01]. The use cases are in German; special characters were removed prior to the clustering. This set contains 66 use cases.

- *Set 2: AUTO.* Here, a control application in the automotive industry is described by use cases. The original specification on which the use case set is based was provided by a very large, international car manufacturer. The use case description was developed at a large German software engineering institute. This set contains 20 use cases (in English).

- *Set 3: COLLAB.* This is a large collaboration and document management framework. Project partners in this joint effort included an international company that is a large producer of white goods, and a very large international company in the market of global telecommunication systems and equipment. This set contains 26 use cases (in English).
- *Set 4: MOBILE.* This large use case set describes the behavior of software functionality in the field of mobile communication. It was created at one of the largest international organizations operating in communication, power infrastructure, and electrical engineering (not the company involved in set 3). This set contains 448 use cases (in English).

The use cases were extracted from the requirement documents and converted (from doc or pdf) to plain-text txt-files. Introductions, use case diagrams etc. were omitted. Table 6.1 gives an overview of the four sets, along with average use case size, and variance of use case size in characters.

6.5 Results

This section presents the maps generated by the self-organizing map approach, and gives the parameters used.

Figure 6.1 represents the map of the TICKET use case set. The sets' identifiers are abbreviations of their content, e.g., "Suc" stands for "Suche" ("search"). Visibly, similar abbreviations are clustered together. It is important to note that these identifiers or abbreviations are not part of the use cases' text, and thus are not analyzed by the

Name	Domain	Number of use cases	Avg. use case size, in characters	Std. dev. use case size
TICKET	Administration	66	1396	460
AUTO	Automotive industry	20	1054	365
COLLAB	Collaboration, document management	26	657	244
MOBILE	Mobile communication	448	1814	642

Table 6.1: Properties of use case sets

	AUTO	COLLAB	TICKET	MOBILE
SOM mode	static	static	static	GHSOM
Dimensions	163	106	313	466
Lower limit	0.05	0.09	0.05	0.03
Upper limit	0.7	0.5	0.7	0.7
Map size	3x3	4x4	5x5	N/A
Iterations	15000	15000	50000	N/A
MQE	6.48	3.03	8.7	N/A

Table 6.2: SOM parameters

SOM; they merely serve to identify use cases and to assess the result of the clustering process.

Figures 6.2 and 6.3 represent the smaller use case set AUTO and COLLAB, respectively. Here, we modified the identifiers to ensure compliance with the confidentiality agreements.

Table 6.2 gives an overview of the parameters used:

- "SOM mode" describes, whether a static or growing hierarchical SOM was used; the latter is particularly useful for very large document sets and was thus used for the MOBILE use case set.

- "Dimensions" gives the number of words used to index the use cases.

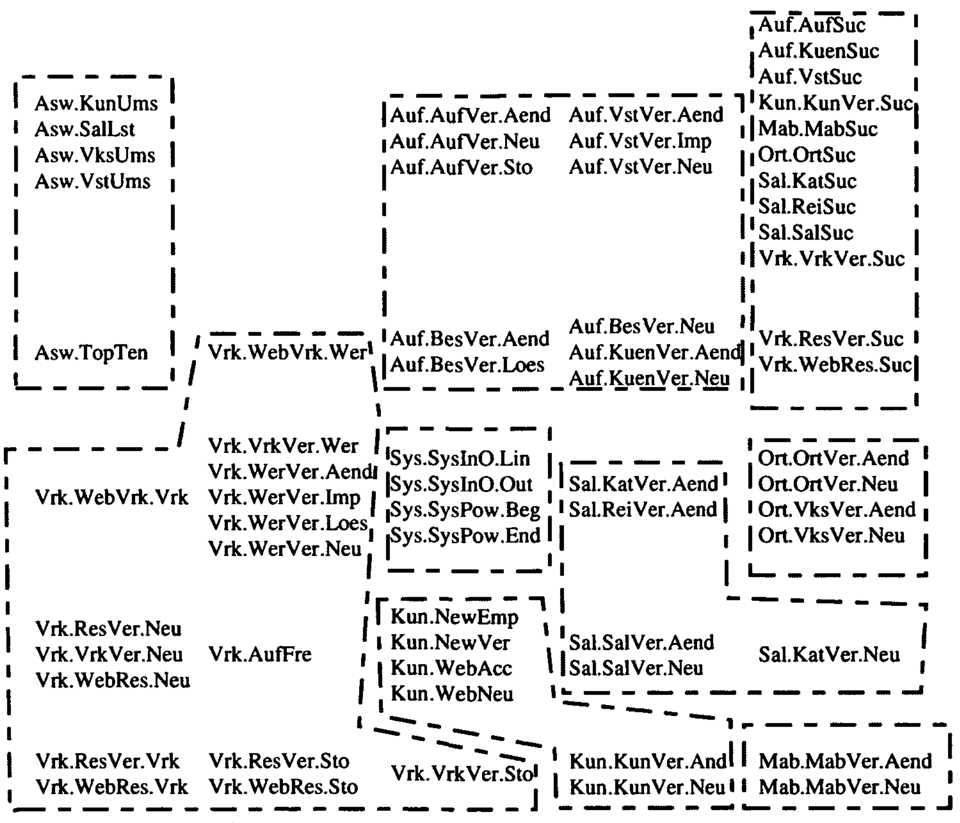

Figure 6.1: Map of use case set TICKET

- "Lower/upper limit" are the %-limits for automatic stop-word definition: if a word appears in too few documents ("lower limit"), or in too many ("upper limit"), it is not included in the set of indexed words.
- "Map size" is the grid size of the 2-dimensional map.
- "Iterations" gives the number of iterations used to generate the map.
- "MQE" is the mean quantization error of the cells contained in a map. The quantization error of a cell measures the dissimilarity of all input data mapped onto a particular unit; the comparison

metric is Euclidean [RMD02].

In all maps, the identified clusters were marked manually.

6.6 Discussion

Analogy-based methods rely on similarities between projects expressed as distances between (often interdependent) attribute sets. Humans, however, are not particularly good at analyzing complex data without the aid of visualization techniques. Thus common tools supporting analogy-based methods, e.g., spreadsheet applications, offer very limited benefit.

Thus, this thesis proposed a transparent way of visualizing the similarity of use cases without the need of explicit measurement data collection; an implicit analogy metric is obtained by using textual similarity.

But what is the difference to merely using a standard text indexing and analyzing the nearest neighbors of a given use case? The 2-dimensional representation of the use case set TICKET can illustrate several key benefits of the graphical clustering.

Using merely the most similar text documents does not give an impression of a particular cluster size. On the upper left, or on the lower right of figure 6.1, for example, relatively small clusters of use cases can be found. On the other hand, on the upper right, a larger number of use cases is clustered together. In the use case set AUTO given in figure 6.2 most clusters (except the 4-use case cluster at the top of the map) consist of two use cases. This becomes immediately visible on the map, but remains invisible using mere text indexing and a list of the most similar documents. This information influences how many neighboring use cases should reasonably be included in the analysis.

A special case are surely isolated instances of use cases, which differ that much from others that they do not appear in any cluster. For example, denoting the lower left cell in figure 6.1 with $(0,0)$, such use cases can be found at $(1,1)$, $(4,1)$, or $(1,3)$ in the TICKET use case set, or on the upper left part of the COLLAB use case set given in figure 6.3. Analyzing such use cases—in our case, estimating related costs using cost information of similar/near use cases—should be done with particular attention as the most similar use cases are already significantly different and their cost information may thus be misleading. Again, a conventional list of the nearest documents would not yield this important information of a use case being isolated.

Another interesting aspect is that the main entity of a use case (denoted by the first 3 characters of the use case identifier, e.g., customer, show, system, etc.) is not determining the clustering. For example, the text describing the procedure of searching for entities (identified by the suffix "Suc") exhibits more textual similarities, overriding the entities' identification similarity and yielding the cluster on the upper right. The search logic was described in greater detail; it seems plausible that estimating a new search use case should analyze other entities' search use cases.

The benefits of visualizing portfolio data in the proposed way could be categorized as follows:

- *Efficient measurement.* No costly consistent collection of project data according to predefined explicit metrics is necessary; instead, artifacts arising during the regular course of requirements analysis together with implicit analogy metrics are used.

- *Efficient to learn.* The proposed method is straightforward and transparent; even estimators not acquainted with it can immediately grasp the process and the natural visualizations' implications.

- *Efficient to use.* In practice, one often has to deal with a yet more or less unknown project structure. This leads to one of the main strengths of the SOM [RMD02]: it gives the user a natural visualization and allows him or her to build a mental model of the underlying project structure facilitating convenient exploration of (probably unknown) past project data.

  For example, in the MOBILE project consisting of more than 440 use cases, the growing hierarchical self-organizing map provides the user with a roadmap witch separates main topical branches, guiding him or her through the various topical sections. It thus substantially reduces the effort needed to gain an overview of the internal project structure and allows comfortable browsing of requirements hierarchies.

The authors are also aware of several instances in industrial environments where estimation was hindered by its relation to software measurement being perceived differently by various stakeholders—by using self-organizing maps to visualize implicitly collected clusters of requirements, the connection between demands and effort becomes transparent, and estimates are easier to agree upon.

The benefits are put in a larger context in chapter 7.

6.7 Conclusion and Outlook

This thesis presented the application of self-organizing maps to specific requirement artifacts, use cases, in order to implicitly determine analogies between new and historic software requirements for software cost estimation. The main motivation was to avoid costly explicit software measurement.

The approach was tested on several real-world industrial use case sets. The clusters obtained did indeed conform to our expectations:

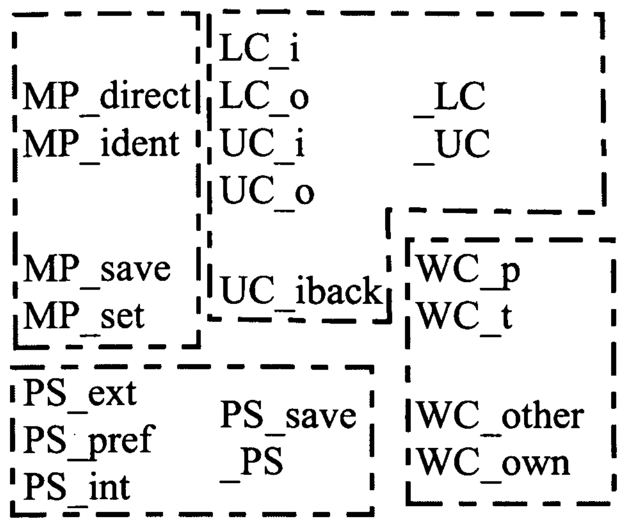

Figure 6.2: Map of use case set AUTO

use cases dealing with similar topics or aspects of a projects were visibly clustered together.

In our view, this approach should be particularly helpful within a well-defined environment, for example, a company, which performs a large number of similar projects in comparable circumstances. Using the approach on singular, innovative projects, or comparing projects of different environments are unlikely to yield satisfactory results.

Further research will define the overall estimation process using this approach in greater detail, and describe tool enhancements to support it.

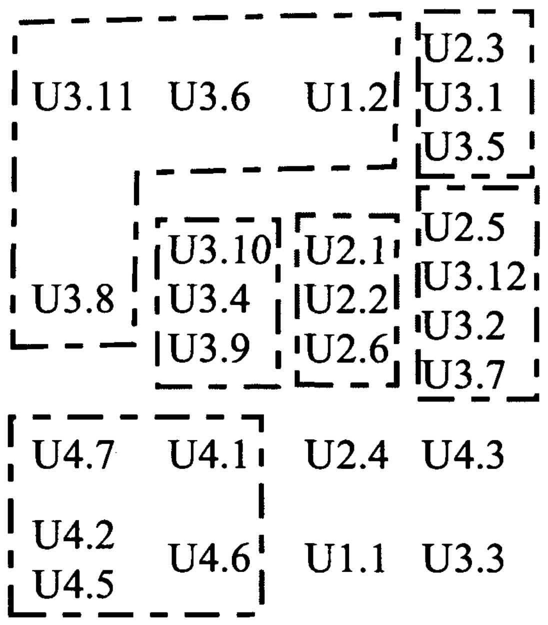

Figure 6.3: Map of use case set COLLAB

Chapter 7

Discussion

This chapter summarizes the main results obtained in each optimization approach of analogy-based cost estimation.

7.1 Overview

In chapter 3 the main cost estimation process steps were outlined as: collect measurement data, create estimate proposal, and check of this proposal by human estimators.

In turn, main aspects of those three process steps have been targeted:

1. The *explicability* of estimation data to human estimators has been improved by applying multi-dimensional scaling methods to the data.

2. The *effectiveness* of the estimates has been improved using extensive project feature weighting.

3. Finally, the *efficiency* of finding suitable analogies in historical project information has been increased by using self-organizing

maps to find textually similar requirement documents (use cases).

Approaches 1 and 2 were further analyzed by quantifying the achieved improvements. With the explicability improvements of MDS visualization, Kruskal's stress value was used as a measure for the approximation quality. Analyzing the data sets suggests to use the most important project feature dimensions (determined by minimal estimation error) to achieve a high-quality approximation even in the two-dimensional visualization.

The effectiveness improvements during estimate proposal creation were measured using standard estimation error measures. Furthermore, a barrier of optimal estimation accuracy is indicated.

All three approaches were tested on real-world, industrial project portfolio data sets. Approaches 1 and 2 were applied to five explicitly collected portfolio data sets available in the public domain. Approach 3—the efficiency of finding analogies—was tested on four real-world use case data sets.

The following sections sum up the results.

7.2 Explicability of Estimates

Although there are many different approaches to support people in estimating software project efforts (e.g., formal models like COCOMO 2, neural networks, regression analysis, etc.), few of them are actually applied in typical industrial environments. Several reasons can be identified—software projects typically involve substantial parts with new and unknown technologies and tools; often, the relevant constraints to a development project is quality rather than effort, and deadlines can be influenced by corporate politics as much as by precise estimations; not to forget, estimation needs to rely on measurement data which is costly and time-consuming to obtain.

However, one important reason is certainly that many proposed methods lack of transparency and accessibility. Especially methods like neural networks give little insight on how they reach a certain estimate and do little to foster portfolio measurement data understanding.

But even seemingly simpler methods like analogy-based approaches can be improved in providing human estimators with context information. Analogy-based methods rely on similarities between projects expressed as distances between high-dimensional features or attribute sets. Humans, however, are not particularly good at analyzing high-dimensional data without the aid of visualization techniques. Thus, simple tools supporting analogy-based methods like spreadsheet applications are severely delimited. Even dedicated tools like ArchANGEL offer only slightly better results—e.g., they relieve the burden of time-consuming and error-prone tasks like normalizing the measurement data—but their result is again a list of projects/feature sets. The degree of the projects' similarities, as well as the structure of the project clusters and thus valuable addition information is not given.

This thesis proposed to enhance analogy-based approaches by visualizing high-dimensional portfolio measurement data with multidimensional scaling. In many circumstances, this is a feasible method to reproduce high-dimensional feature sets graphically; the approximation quality can be measured by the stress value. Data sets with 6 and more dimensions were visualized successfully within reasonable stress boundaries given in [Kru64a].

The benefits of visualizing portfolio data are manifold:

- *Transparency.* The proposed method is straightforward and transparent; even estimators not acquainted with it immediately grasp the process and the visualizations' implications. We are aware of several instances in industrial environments where estimation was hindered by its relation to software measurement being perceived differently by various stakeholders—by applying MDS to multidimensional data, the connection between metrics and result becomes transparent, and measurement procedures are easier to agree upon. Finally, as no model configuration or difficult-to-reproduce algorithms are involved, users are far more likely to accept and apply this method in the first place.

- *Overview.* MDS gives the user a visualization with a high information density. It is therefore easy to gain a fast overview of a project portfolio's properties, for example, its project cluster structures and sizes. If based on the same metrics, the method allows for a fast comparison of different portfolios—the portfolios' entropy properties are visualized in a highly intuitive way.

  For example, while the projects in the Desharnais 2 data set form some clusters (see figure 4.5), the projects in the Desharnais 3 data set are less coupled.

- *Methodology.* Several publications comparing estimation methods indicate that no method can generally be regarded as the best one; a method's performance depends highly on the underlying portfolio data properties. Visualizing the data can help estimators to assess whether it is reasonable to apply analogy-based methods in a specific circumstance or whether a particular project cluster structure is unlikely to yield high-quality analogy-based estimates. This could happen if the project to be estimated is distant to relevant project clusters, if the nearest cluster is very small, or if the effort variance in the nearest cluster is too high. In that case, other methods, like regression analysis, could be used to overrule the analogy-based estimate.

  For example, projects 6 and 10 in the Desharnais 3 data set (see figure 4.8) should probably not be estimated using the analogy-based approach as they are distant to the rest of the projects.

- *Operation.* The task of analyzing analogies in portfolio data involves identifying similar project feature sets. This can be performed fast and reliably on a visual representation of the data, especially as the criteria are varying (e.g., in some cases a larger cluster could be used as basis for the estimate if it is dense, while in other cases instead of a fixed number of similar projects only one or few should be used due to a portfolio's high entropy). Outliers, which can degrade the estimate's quality considerably, can be identified and removed easily—both projects that are distant to all other projects, and projects that are within a cluster but behave differently with regard to the related effort value. It would be possible to enhance conventional tools to perform similar tasks, for example, by making them configurable using threshold values for distances and cluster homogeneity, but this would make the tool far less transparent and accessible.

For example, project 5 in the Albrecht data set (see figure 4.2) should probably not be allowed to influence estimates of nearby projects—its high effort value should first be analyzed to decide if this is a valid project to compare other projects to.

- *Confidence.* Finally, the benefits mentioned above (method transparency and user acceptance; coarse portfolio overview and understanding; assessment of a methodology's suitability; easy data selection and manipulation) contribute to increase the confidence in a particular estimation. Usually, estimation methods were compared using accuracy and reliability measures; they did not take into account the confidence an estimator had in its estimate at the time of estimation. The transparency of the proposed visual support is likely to increase this confidence, which should allow—in many cases—to agree on more narrow estimates.

  For example, the lower right project cluster of the Desharnais 2 data set (see figure 4.5) seems—despite some outliers—to increase confidence in an effort estimate range between 2500 and 3500.

7.3 Effectiveness of Estimation

High-quality—i.e., accurate and reliable—estimates are fundamental to many success-critical portfolio decisions. This thesis aims at improving conventional analogy-based techniques by addressing the varying impact of different project features.

The following questions concerning estimation accuracy, reliability and volatility were raised:

1.a Is the *MMRE* value *decreased* significantly, indicating higher estimation accuracy?

1.b Using the *MMRE*-optimal dimension weights, is another accuracy indicator, namely, $Pred_{25}$ *increased*, thus further supporting a higher estimation accuracy?

2. Is the variance of the relative error (denoted with Var_r) *decreased*, indicating higher estimation reliability?

3. Is the volatility measure *MMDWC* decreased, indicating a more stable model?

Accuracy. Using more than 2 dimension weight values, the *MMRE* value was decreased for all 5 portfolio data sets, up to 24%. Applying the new approach to growing portfolios, i.e., portfolios created by adding new projects to an initial portfolio, further supports this impression—the median *MMRE* reduction was up to 22% for the Desharnais 3 data set.

Another commonly used measure for estimation accuracy is $Pred_{25}$. Using the *MMRE*-optimal dimension weights, a portfolio's $Pred_{25}$ value was calculated to find out whether it would support the impression of higher estimation accuracy indicated by the primary measure, *MMRE*. It is shown that *MMRE*-optimal dimension weights usually yield a better $Pred_{25}$ performance measures, too: both measures support the impression of improved estimation accuracy.

Reliability. The variance of the relative error Var_r can be used as a measure to indicate an estimation method's reliability. Figure 5.5 illustrates, that the more dimension weight values are used, the more the Var_r value is improved, i.e., reduced.

Volatility. As pointed out in section 2, conventional dimension weighting (i.e., selection) is very sensitive to outliers, which frequently results in oscillating weight values when adding new projects to portfolios. Figure 5.6 illustrates how weight values change when new projects are added to the Desharnais 2 data set until the set is complete. Such oscillating weights are counterintuitive and make estimate proposals more difficult to re-enact for the estimator; in addition, it undermines the confidence in estimate proposals as the weight changes seem rather arbitrary.

In order to quantify this volatility, a simple measure—as used in time series analysis—was introduced in section XXX, the mean magnitude of dimension weight changes *MMDWC*. For all data sets, this measure is reduced substantially, up to 65%.

This is particularly interesting as it indicates—much clearer than the previous, binary approach—that some features are really more important than other even in changing, i.e., growing, portfolios. There influence could be described as continuous, whereas the binary switches rather point to arbitrary ad-hoc-correlations.

Thus, the questions can be answered: yes, *MMRE* is reduced, $Pred_{25}$ is increased, both indicating a higher estimation accuracy; yes, Var_r is reduced, indicating higher estimation reliability; yes, *MMDWC* is reduced, indicating a more stable and less volatile method.

However, while we advocate to use measures such as *MMRE* cautiously, to compare methods only within the same portfolio, and not to derive conclusions from the measures' absolute value, another important result can be derived from the magnitude of the estimation quality measure improvements: when using the strict equals-relation in the definition of the set of similar projects (see section 3), and the Euclidean distance, even human estimators can not find better dimension weight values with respect to the traditional measures like *MMRE*. Thus, further improvement must focus on other aspects, like removing outliers in the portfolio, or selecting relevant subsets of projects.

The achieved estimation quality improvements notwithstanding, it is important to note that the optimization target was one specific stage of the overall estimation process, and that good estimation is not achieved but supported by formal methods like the approach presented in this thesis.

7.4 Efficiency of Estimation

Although there are many different approaches to support people in estimating software project efforts, few of them are actually applied in typical industrial environments. One of the main reasons clearly is that estimation usually needs to rely on explicitly collected measurement data which is costly and time-consuming to collect.

Moreover, many estimation methods lack of transparency and accessibility. Black-box approaches methods, e.g., relying on neural nets, often provide little insight on how they reach a certain estimate, and do little to foster portfolio measurement data understanding.

Thus, this thesis proposed a transparent way of visualizing the similarity of use cases without the need of explicit measurement data collection; an implicit analogy metric is obtained by using textual similarity.

This is achieved using self-organizing maps, which have a strong tradition in the text mining area. They are a very powerful tool for detecting structure in high-dimensional data and organizing it accordingly on a two-dimensional output space. Several of its proven properties seem to be highly useful and supportive in the context of project portfolio cost estimation and decision support, where users should benefit particularly from clustering techniques that uncover similar documents and bring these similarities to the user's attention, allowing for: the co-location of similar, yet not identical topics; the content-based organization to facilitate browsing of documents according to subject hierarchies; the topology-preserving representation of input similarities in terms of distances in the output space; and the ease by which a user gains an idea regarding the structure of the underlying data by analyzing the map [RM03].

The general benefits of visualizing portfolio data in the proposed way could be categorized as follows:

- *Efficient measurement.* No costly consistent collection of project data according to predefined explicit metrics is necessary; instead, artifacts arising during the regular course of requirements analysis together with implicit analogy metrics are used. Thus applying the proposed method is highly efficient: all that is required to visualize a set of use cases is to extract them from existing documents and find optimal parameter settings. Costs and complexity, and thus also the inhibition level of introducing the method into practice, are substantially reduced.

- *Efficient to learn.* The proposed method is straightforward and transparent; even estimators not acquainted with it can immediately grasp the process and the natural visualizations' implications.

  Moreover, as no definition of metrics, model configuration or difficult-to-reproduce algorithms are involved, users are far more likely to accept and apply this method in the first place.

- *Efficient to use.* In practice, one often has to deal with a yet more or less unknown project structure. This leads to one of the main strengths of the SOM [RMD02]: it gives the user a natural visualization and allows him or her to build a mental model of the underlying project structure facilitating convenient exploration of (probably unknown) past project data. Therefore, orientation for the estimator is highly improved, and it is easy to gain a fast overview of a project's properties, its functionality's cluster structures and sizes. This applies for team members new to a project as well as experienced managers and estimators tackling a new project.

These properties of the proposed visual representation—transparency and simplicity, overview and understanding—also seem to be of particular relevance for industrial practice as expert judgment based on informal analogies between projects is by far the most frequently applied estimation technique [HHA91].

A company using this approach to quickly find similar past requirements in order to estimate new ones should implement a requirement database; manual document handling and conversion from doc to txt file formats would nullify efficiency gains by this implicit analogy-based approach.

If use cases are written by different people, their language and wording could influence the clustering; therefore important terms should be defined and managed centrally. (This is helpful to obtain consistent requirements, anyway.)

If use case templates are used, which contain sections like, e.g., "precondition", empty sections should not be removed. Thus, the section name will be correctly identified as stop-word and will not influence the clustering. Authors' names, however, should be removed from the use case text, potentially requiring a change in the use case template and database.

To keep the system as flexible and generally applicable as possible, no language- or domain-specific optimizations for the content-based analysis, such as the use of specific stop-word lists, were performed. Language-specific adaptation may further improve the resulting clustering, albeit sacrificing the language and domain independence [RMD02].

7.5 Summary

The main focus of our approach was to improve the (a) explicability of estimation when estimators check the proposals, the (b) effectiveness, i.e., the quality of estimation with regard to common error metrics, and the (c) efficiency of determining suitable analogies with historical portfolio information.

This was achieved using MDS visualization, extensive feature weighting, and SOM maps, respectively. Qualitative metrics are used to determine the improvement for (a) and (b); in all three cases, real-world portfolio information was used to assess the methods applicability.

However, the improvements are not restricted to the process step they originally aimed at. The extensive feature weighting approach (b), for example, substantially reduces the volatility of feature weights, increasing the acceptability (and thus the explicability) of the analogy-based model.

The MDS visualization (a) not only increased explicability and transparency, but made the cumbersome analysis of high-dimensional data much more efficient.

All improvements should lower the barrier for an implementation of formal analogy-based estimation: this would constitute an important estimation effectiveness improvement in real-world environments.

Chapter 8

Summary and Further Work

This chapter briefly outlines further research in the areas and approaches presented in the previous chapters. This follow-up work will mostly concentrate on extending the analogy-based approach to neighboring fields, e.g., risk assessment, supporting the proposed methods with tools, and analyzing more and larger data sets than those available at the time of writing.

8.1 Integrating Visualization/Further Data

Multi-dimensional scaling (MDS) methods were used to visualize portfolio measurement data. Using a selection of the most important project features, a very high degree of approximation quality could be achieved on several real-world data sets. However, this prototypical procedure currently involves several different tools, which makes its use somehow cumbersome to inexperienced users.

Future research should provide a more integrated user experience, including the MDS and the preparation steps necessary seamlessly into the measurement or estimation environment. One possibility that might be explored is to provide a Web-based access to the MDS visualization.

In addition to mere numerical information given on the MDS visualization, qualitative project information, links to the responsible stakeholders or key documents, etc., could be displayed.

Another aspect is to extend MDS to nominal measurement values. This could include using sub-MDS to define dissimilarities between single dimensions of a project feature tuple. This would open MDS analysis for well-known project portfolio data sets like the ESA and Laturi sets [Wie01].

8.2 Extending Feature Weighting

The approach of extensively weighting project features can be extended in several directions, for example, in using more than one similar project to determine the estimate (effectively smoothing the effect of outliers on the estimation quality).

Secondly, nominal values—currently not supported by our tool AMBER—will be covered, allowing for an analysis of several ESA and Laturi data sets [Wie01]. The current version of AMBER is already extended to support nominal values; together with Fraunhofer Institute's Adam Trendowicz we are analyzing data sets containing several nominal feature dimensions.

Out tool AMBER currently concentrates on one estimation quality metric, and provides two secondary metrics which should assess the overall quality improvements. The latter ones, however, can't yet be used as target metrics, i.e., as stop criteria for optimal feature weights. Future versions should include this ability.

Finally, this approach should be integrated with the MDS preparation step, thus further improving the MDS approximation quality.

8.3 Further Settings for Implicit Metrics

The key benefit of the self-organizing maps used to implicitly determine textual similarities of use case for cost estimation was that no expensive explicit measurement program is required; similar past project documents are found because they have similar textual characteristics.

In our case, use cases were used as document artifacts; they are comparable in size and scope, and usually well-defined templates guarantee their consistency.

If similar artifacts can be identified in other fields, especially risk assessment, this analogy-based approach could be extended to cover other knowledge re-use applications in the wide field of project portfolio decision making.

Again, tool integration, for example, with requirement database systems, is necessary to achieve acceptability in a real-world industrial environment. Future development would certainly encompass open interfaces to support data exchange with use case repositories.

Another focus could lie on artifacts created in new, agile process environments, like user stories. Difficulties could arise if stakeholder would feel restrictions imposed by inflexible tools—thus, the tool integration mentioned above would probably have to precede this.

Appendix A

Use Case Samples

To illustrate the SOM approach used to cluster textually similar use case, several use case from the use case set TICKET are given here. The use cases are in German. The other three use case sets analyzed in this thesis can't be disclosed because of confidentiality agreements with the respective companies.

For each main entity of the TICKET use case set, one example is given.

```
Titel: Auswertung Auslastung der Saele
Kurzbeschreibung: Ermittelt alle Saele, die bestimmten
Kriterien entsprechen und sortiert diese nach Auslastung.
(Ausgabe von: Bezeichnung, Typ, AnzPlaetze und die
berechnete Auslastung.)
Vorbedingungen: Anwender mit Berechtigung Marketing
oder hoeher ist eingeloggt.
Beschreibung des Ablaufes:
E1) Der Anwender zeigt an, die Auslastung der Saele
abfragen zu wollen.
A1) Das System ueberprueft, ob Verbindung zur
Datenbank besteht.
E2) Datenbankverbindung steht.
A2) Listenfelder fuer Auswahlkriterien werden mit
vorhandenen Werten der Tabellen gefuellt.
AE2) Datenbankverbindung unterbrochen.
AA2) Fehlermeldung ausgeben.
E3) Der Anwender gibt die Auswahlkriterien ein, nach
```

denen die Auswertung eingeschraenkt werden soll
(Zeit_von, Zeit_bis, Kategorie, Auffuehrungsort, Ort,
relative/absolute Darstellung, ...) und startet die
Abfrage.
A3) Das System sucht nach Saelen, die den angegebenen
Kriterien entsprechen.
E4) Es existieren Datensaetze, die den angegebenen
Kriterien entsprechen.
A4) Die gefundenen Datensaetze werden als Liste und
als Grafik angezeigt.
AE4) Es konnte kein Datensatz gefunden werden, der
den angegebenen Kriterien entspricht.
AA4) Hinweismeldung ausgeben.
E5) Der Anwender zeigt an, das Ergebnis ausdrucken
zu wollen.
A5) Das Ergebnis wird als Liste und als Grafik am
Drucker ausgegeben.
E6) Der Anwender zeigt an, das Ergebnis als Tabelle
speichern zu wollen und gibt einen Dateinamen an.
A6) Die Daten werden exportiert. (z.B. Text mit Tab,
Excel, ...)
Auswirkungen: keine
Nichtfunktionale Anforderungen: keine
Anmerkungen: Welche Parameter zur Einschraenkung der
Suchkriterien tatsaechlich herangezogen werden sollen,
bleibt noch mit der Marketing-Abteilung zu klaeren.
Ad E3) eventuell waere es hier auch moeglich aus
verschiedenen Charts auszuwaehlen.
Ad E6, A6) optional mit Marketing klaeren.

Titel: Auffuehrung stornieren
Kurzbeschreibung: Storniert Auffuehrungen bzw. sagt diese
ab.
Vorbedingungen: Anwender mit einer Berechtigung Vadm oder
hoeher ist eingeloggt. Es wurde bereits eine Auffuehrung
angelegt.
Beschreibung des Ablaufes:
E1) Der Anwender zeigt an, eine Auffuehrung stornieren zu
wollen.
A1) Der Anwendungsfall "Auffuehrung suchen" (AufSuc) wird

ausgefuehrt.
E2) Die Suche nach einer Auffuehrung war erfolgreich und eine Auffuehrung wurde ausgewaehlt.
A2) Eine Warnung wird ausgegeben.
AE2) Es wurde keine Auffuehrung gefunden, oder die Suche wurde abgebrochen.
AA2) Fehlermeldung ausgeben.
E3) Der Anwender bestaetigt die Auffuehrung stornieren zu wollen.
A3) Die Auffuehrung wird storniert und alle zugewiesenen Reservierungen ebenfalls (siehe Anmerkungen).
AE3) Der Anwender bricht die Stornierung ab.
AA3) Hinweis ausgeben.
Auswirkungen: Es wird eine Auffuehrung storniert.
Nichtfunktionale Anforderungen: keine
Anmerkungen: Sollten bereits Reservierungen fuer diese Auffuehrung existieren, so sind diese zu stornieren. Verkaeufe werden nicht automatisch storniert. Es ist hierbei vorgesehen, dass die entsprechenden Kunden persoenlich zu einer Verkaufsstelle kommen und stornieren lassen.

Titel: Kunde anlegen
Kurzbeschreibung: Ein Kunde mit allen zugehoerigen Daten (KartenNr, Mitarbeiter, NName, VName, Titel, Geschlecht, Geburtsdatum, Strasse, PLZ, Ort, TelNr, TicketCardGruppe, KontoNr, Kontostand, Ermaeigung, GueltigBis, Gesperrt, PIN und Vorlieben) wird angelegt.
Vorbedingungen: Anwender mit einer Berechtigung Verkauf oder hoeher ist eingeloggt.
Beschreibung des Ablaufes:
E1) Der Anwender gibt die Daten des neuen Kunden ein.
A1) Das System ueberprueft, ob alle erforderlichen Daten eingegeben wurden, die Daten korrekt sind und der Kunde noch nicht im System vorhanden ist.
E2) Alle erforderlichen Daten wurden eingegeben, sind korrekt und der neue Kunde ist noch nicht im System vorhanden.
A2) Der Anwendungsfall "Ort suchen" (Ort.OrtSuc) wird ausgefuehrt.
AE2) Es wurden nicht alle erforderlichen Daten eingegeben,

```
sie sind nicht korrekt oder  der neue Kunde ist bereits im
System vorhanden.
AA2) Der Anwendungsfall wird abgebrochen.
E3) Eine Verkaufsstelle wurde gefunden und ausgewaehlt.
A3) Der Kunde wird der Verkaufsstelle zugeordnet.
AE3) Es wurde keine Verkaufsstelle gefunden oder
ausgewaehlt.
AA3) Der Anwendungsfall wird abgebrochen.
E4) Der Kunde ist einer Verkaufsstelle zugeordnet.
A4) Die Daten des Kunden werden abgespeichert.
Auswirkungen: Ein neuer Kunde wurde angelegt und einer
Verkaufsstelle zugeordnet.
Nichtfunktionale Anforderungen: keine
Anmerkungen: Zum Neuanlegen von Kundendaten zaehlt auch
das Hinzufuegen von Daten bezueglich Web-Account.


Titel: Mitarbeiter anlegen
Kurzbeschreibung: Ein Mitarbeiter mit allen zugehoerigen
Daten (KartenNr, NName, VName, Titel, Ge-schlecht,
Geburtsdatum, Strasse, PLZ, Ort, TelNr, Berechtigung und
Passwort) wird angelegt und einer Verkaufsstelle
zugewiesen.
Vorbedingungen: Anwender mit einer Berechtigung
Systemtechniker ist eingeloggt. Verkaufsstelle existiert.
Beschreibung des Ablaufes:
E1) Der Anwender gibt die Daten des neuen Mitarbeiters ein.
A1) Das System ueberprueft, ob alle erforderlichen Daten
eingegeben wurden, die Daten korrekt sind und der
Mitarbeiter noch nicht im System vorhanden ist.
E2) Alle erforderlichen Daten wurden eingegeben, sind
korrekt und der neue Mitarbeiter ist noch nicht im System
vorhanden.
A2) Der Anwendungsfall "Ort suchen" (Ort.OrtSuc) wird
ausgefuehrt.
AE2) Es wurden nicht alle erforderlichen Daten eingegeben,
sie sind nicht korrekt oder der neue Mitarbeiter ist bereits
im System vorhanden.
AA2) Der Anwendungsfall wird abgebrochen.
E3) Eine Verkaufsstelle wurde gefunden und ausgewaehlt.
A3) Der Mitarbeiter wird der Verkaufsstelle zugeordnet.
```

AE3) Es wurde keine Verkaufsstelle gefunden oder ausgewaehlt.
AA3) Der Anwendungsfall wird abgebrochen.
E4) Der Mitarbeiter ist einer Verkaufsstelle zugeordnet.
A4) Die Daten des Mitarbeiters werden abgespeichert.
Auswirkungen: Ein neuer Mitarbeiter wurde angelegt und einer Verkaufsstelle zugeordnet.
Nichtfunktionale Anforderungen: keine
Anmerkungen: Um beim ersten Start des Programms Zugriff auf vor allem diesen und ueberhaupt alle Anwendungsfaelle, die einen eingeloggten Anwender voraussetzen, zu haben, muss ein UrAnwender (mit den Rechten eines Systemtechnikers ausgestattet) existieren. Dieser Anwender ist automatisch so lange aktiv, bis der erste Systemtechniker angelegt wurde. Sobald also ein Systemtechniker als Anwender erstellt wurde, kann dieser UrAnwender nicht mehr am System angemeldet werden.

Titel: Ort anlegen
Kurzbeschreibung: Ein Auffuehrungsort mit allen Daten (Bezeichnung, Ortsname, Strasse, PLZ, Bundesland, Oeffnungszeiten, TelNr und Besitzer sowie die Flag Auffuehrungsort) wird angelegt.
Vorbedingungen: Anwender mit einer Berechtigung Vadm oder hoeher ist eingeloggt.
Beschreibung des Ablaufes:
E1) Der Anwender gibt die Daten des neuen Ortes ein.
A1) Das System ueberprueft, ob alle erforderlichen Daten eingegeben wurden, die Daten korrekt sind und der Ort noch nicht im System vorhanden ist.
E2) Alle erforderlichen Daten wurden eingegeben, sind korrekt und der neue Ort ist noch nicht im System vorhanden.
A2) Die Daten des neuen Ortes werden abgespeichert.
AE2) Nicht alle erforderlichen Daten eingegeben bzw. nicht korrekt oder der Ort ist bereits im System vorhanden.
AA2) Fehlermeldung ausgeben.
Auswirkungen: Ein neuer Ort wurde angelegt und abgespeichert.
Nichtfunktionale Anforderungen: keine
Anmerkungen: keine

Titel: Saal aendern
Kurzbeschreibung: Die Daten eines vorhandenen Saals werden
geaendert.
Vorbedingungen: Anwender mit einer Berechtigung Vadm oder
hoeher ist eingeloggt. Ein Saal wurde bereits angelegt.
Beschreibung des Ablaufes:
E1) Der Anwender zeigt an, einen Saal aendern zu wollen.
A1) Der Anwendungsfall "Saal suchen" (Sal.SalSuc) wird
ausgefuehrt.
E2) Die Suche nach einem Saal war erfolgreich und ein Saal
wurde ausgewaehlt.
A2) Der entsprechende Saal und dessen Daten werden zur
Bearbeitung angezeigt (inklusive dem zugewiesenen
Auffuehrungsort und den untergeordneten Kategorien in
Listenform).
AE2) Es wurde kein Saal gefunden, oder die Suche wurde
abgebrochen.
AA2) Der Anwendungsfall wird abgebrochen.
E3) Der Anwender aendert die Daten des Saales (siehe
Anmerkungen).
A3) Das System ueberprueft, ob keine erforderlichen Daten
fehlen und die Daten korrekt sind.
E4) Alle erforderlichen Daten wurden eingegeben und sind
korrekt.
A4) Die geaenderten Daten des Saales werden abgespeichert.
AE4) Nicht alle erforderlichen Daten wurden eingegeben bzw.
sind nicht korrekt oder die Eindeutigkeit der Daten ist
nicht mehr gewaehrleistet.
AA4) Fehlermeldung ausgeben und Sprung zu E3.
Auswirkungen: Die Daten eines Saales wurden geaendert
und abgespeichert.
Nichtfunktionale Anforderungen: keine
Anmerkungen: Das Zuweisen eines anderen Auffuehrungsortes
ist ebenfalls als Aenderung der Daten eines Saales zu
verstehen und erfolgt mit Hilfe des Anwendungsfalles
"Ort suchen" (Ort.OrtSuc). Ebenso ist das Hinzufuegen oder
Aendern von Kategorien als Aenderung der Daten eines Saales
zu verstehen. Diese Funktionalitaet ist durch einen Aufruf
der Anwendungsfaelle "Kategorie anlegen" (KatVer.Neu) bzw.
"Kategorie aendern" (KatVer.And) gewaehrleistet. Ersterem
wird der bearbeitete Saal als Parameter uebergeben. Dem

Anwendungsfall "Kategorie aendern" (KatVer.And) wird die
vom Anwender aus der Liste ausgewaehlte Kategorie
uebergeben.

Titel: Hochfahren des Pogramms.
Kurzbeschreibung: Das Programm wird gestartet und wartet
anschlieend auf einen Login.
Vorbedingung: Das Programm laeuft nicht. Das Programm ist
korrekt installiert. (Ueberpruefung der Vollstaendigkeit
der wesentlichen Programmteile).
Beschreibung des Ablaufs:
E1) Der Anwender startet das Programm wie im verwendeten
System ueblich.
A1) Das Programm ueberprueft ob genuegend Ressourcen zur
Verfuegung stehen, das Programm korrekt installiert wurde
(Datenbank, Alias,...) und noch nicht laeuft.
E2) Es sind genuegend Ressourcen vorhanden, das Programm
wurde korrekt installiert und laeuft noch nicht.
A2) Das Programm startet.
AE2) Es sind nicht genuegend Ressourcen vorhanden, das
Programm wurde nicht korrekt installiert oder laeuft bereits.
AA2) Fehlermeldung ausgeben.
E3) Das Programm wurde gestartet.
A3) Der Use-Case "Login eines Anwenders" (SysInO.Lin)
wird ausgefuehrt.
Auswirkungen: Das Programm laeuft und wartet auf einen
Login.
Nichtfunktionale Anforderung: Waehrend des Start des
Programms soll ein Logo zu Werbezwecken erscheinen.
Anmerkungen: Keine.

Titel: Tickets verkaufen
Kurzbeschreibung: Legt eine Transaktion (DatumUhrzeit,
Verkauft, Storniert, Preis, ResNr, Startplatz,
AnzPlaetz, Zahlart) an.
Vorbedingungen: Anwender mit einer Berechtigung
Verkaeufer oder hoeher ist eingeloggt
Beschreibung des Ablaufs:
E1) Der Anwender zeigt an, einen neuen Verkauf durchfuehren

```
zu wollen.
A1) Der Anwendungsfall "Kunden suchen" (Kun.KunVer.Suc)
wird ausgefuehrt.
E2) Ein Kunde wurde gefunden und ausgewaehlt.
A2) Der Anwendungsfall "Auffuehrung suchen" (AufFre)
wird ausgefuehrt.
AE2) Es wurde kein Kunde gefunden oder ausgewaehlt.
AA2) Ein Dummy-Kunde wird fuer den Verkauf herangezogen.
E3) Eine Auffuehrung wurde gefunden und ausgewaehlt.
A3) Es werden alle dem Saal, in dem die Auffuehrung
stattfindet, untergeordneten Kategorien gesucht.
AE3) Es wurde keine Auffuehrung gefunden oder ausgewaehlt.
AA3) Fehlermeldung ausgeben.
E4) Es wurden Kategorien gefunden und der Anwender waehlt
eine davon aus.
A4) Es werden alle zu dieser Kategorie gehoerigen Reihen
und deren Reservierungs- bzw. Verkaufsstatus angezeigt.
AE4) Es wurde keine Kategorie gefunden oder ausgewaehlt.
AA4) Fehlermeldung ausgeben.
E5) Der Anwender waehlt alle zu verkaufenden Plaetze und
bestaetigt den Verkauf.
A5) Die Plaetze werden im System sofort als verkauft
markiert. Sollte noch keine Belegung fuer die gewaehlte
Auffuehrung und Reihe existieren, wird diese ebenfalls
jetzt angelegt.
AE5) Es wurden keine Plaetze fuer den Verkauf gewaehlt.
AA5) Fehlermeldung ausgeben.
E6) Die angegebenen Plaetze wurden verkauft. Es existiert
eine Belegung fuer diese Auffuehrung und Reihe.
A6) Rueckfrage nach weiteren Verkaeufen.
E7) Der Anwender bestaetigt die Rueckfrage nach weiteren
Verkaeufen.
A7) Sprung zu A2.
AE7) Der Anwender lehnt weitere Verkaeufe ab.
AA7) Der Verkauf wird abgeschlossen (siehe Anmerkungen).
Auswirkungen: Es wird ein Kartenverkauf durchgefuehrt und
die entsprechenden Plaetze als verkauft markiert. Die
entsprechende Transaktion wird verspeichert.
Nichtfunktionale Anforderungen: Dies ist einer der
wichtigsten Anwendungsfaelle des gesamten Systems. Eine
uebersichtliche Aufbereitung der Suchergebnisse (Kunde,
```

Veranstaltung,...) ist besonders wichtig. Auch an die Darstellung der reservierten und verkauften Plaetze werden hoechste Ansprueche gestellt, da sie die Grundlage fuer die Beratung des Personals bei Wuenschen des Kunden verkoerpern.
Anmerkungen: Mit dem Abschluss des Verkaufes ist auch das eventuell in spaeteren Versionen durchzufuehrende Ausdrucken des/der Tickets gemeint. Der Anwendungsfall "Kategorie suchen" (Sal.KatSuc) kommt bei A3 bewusst nicht zum Einsatz, da diese Suche automatisch erfolgen soll und ohne weitere Eingaben des Anwenders.

Appendix B

Data Sets

name	effort	in	out	file	inq	fp	sloc
1	10240	25	150	60	75	1750	130
2	10520	193	98	36	70	1902	318
3	1110	70	27	12	0	428	20
4	2110	40	60	12	20	759	54
5	2880	10	69	9	1	431	62
6	1000	13	19	23	0	283	28
7	800	34	14	5	0	205	35
8	490	17	17	5	15	289	30
9	1290	45	64	16	14	680	48
10	1900	40	60	15	20	794	93
11	1080	41	27	5	29	512	57
12	290	33	17	5	8	224	22
13	750	28	41	11	16	417	24
14	1200	43	40	35	20	682	42
15	410	7	12	8	13	209	40
16	1580	28	38	9	24	512	96
17	1830	42	57	5	12	606	40
18	890	27	20	6	24	400	52
19	3810	48	66	50	13	1235	94
20	6120	69	112	39	21	1572	110
21	360	25	28	22	4	500	15
22	1180	61	68	11	0	694	24
23	50	15	15	3	6	199	3
24	610	12	15	15	0	260	29

Table B.1: Albrecht data set

no	effort	team_exp	man_exp	transact	entities	p_a	p_na	enverg	yr_end	length
1	5152	1	4	253	52	305	302	34	85	12
2	5635	0	0	197	124	321	315	33	86	4
3	805	4	4	40	60	100	83	18	85	1
4	3829	0	0	200	119	319	303	30	86	5
5	2149	0	0	140	94	234	208	24	86	4
6	2821	0	0	97	89	186	192	38	86	4
7	3913	1	2	186	52	238	214	25	83	13
8	7854	3	1	172	88	260	247	30	85	12
9	2422	3	4	78	38	116	103	24	83	4
10	4067	4	1	167	99	266	237	24	84	21
11	9051	2	1	146	112	258	271	40	84	17
12	2282	1	1	33	72	105	88	19	84	3
13	4172	3	4	162	61	223	216	32	85	8
14	4977	4	4	223	121	344	320	28	85	9
15	3192	4	3	57	43	100	108	43	85	8
16	840	4	2	58	34	92	86	29	86	5
17	5180	2	4	88	170	258	255	34	85	18
18	5775	2	4	306	132	438	447	37	86	5
19	10577	4	1	304	78	382	397	39	87	20
20	3983	1	4	89	200	289	283	33	86	8
21	3164	4	1	86	230	316	310	33	85	14
22	3542	2	0	71	235	306	312	37	86	6
23	4277	3	1	148	324	472	491	39	85	14
24	7252	4	4	116	170	286	263	27	85	16
25	3948	4	1	175	277	452	461	37	85	14
26	3927	4	3	79	128	207	190	27	86	6
27	6405	1	1	194	91	285	285	35	85	5
28	4620	1	1	451	48	499	464	28	87	9
29	2174	1	1	64	54	118	106	25	88	10
30	6699	2	1	182	126	308	308	35	86	18
31	14987	2	3	173	332	505	424	19	87	27
32	4004	2	2	252	7	259	241	28	88	9
33	12824	4	3	131	180	311	361	51	85	11
34	2331	2	3	106	39	145	103	6	85	8
35	5817	3	3	96	108	204	192	29	85	9
36	2989	2	3	116	72	188	156	18	85	7
37	3136	3	3	86	49	135	131	32	85	6
38	14434	2	3	221	121	342	342	35	85	17
39	2583	1	1	61	96	157	130	18	87	12
40	2520	1	1	78	99	177	140	14	86	5
41	1603	4	7	69	74	143	113	14	86	13
42	2800	4	3	227	73	300	297	34	83	12
43	9520	4	4	395	193	588	617	40	86	24
44	23940	4	4	886	241	1127	1116	34	85	36

Table B.2: Desharnais 1 data set

no	effort	team_exp	man_exp	transact	entities	p_a	p_na	enverg	yr_end	length
1	2569	2	1	119	42	161	145	25	85	9
2	1617	3	2	119	48	167	152	26	85	8
3	3437	4	4	68	316	384	326	20	86	14
4	4494	3	4	9	386	395	340	21	87	14
5	14973	4	4	318	269	587	581	34	86	12
6	9135	1	3	137	119	256	253	34	86	17
7	8050	3	3	302	145	447	523	52	88	16
8	3647	1	3	132	89	221	155	5	86	12
9	8232	3	7	45	387	432	350	16	86	13
10	3276	1	1	55	112	167	129	12	86	12
11	2723	1	4	124	52	176	139	14	87	8
12	3472	3	3	120	126	246	197	15	87	5
13	1575	1	2	47	32	79	62	14	87	6
14	2926	1	1	126	107	233	205	23	86	12
15	1876	3	2	101	45	146	117	15	86	6
16	3626	1	3	194	97	291	291	35	86	8
17	11361	2	4	323	184	507	507	35	87	15
18	1267	1	3	42	31	73	67	27	86	10
19	2548	1	2	74	43	117	105	25	87	5
20	1155	3	4	101	57	158	117	9	87	10
21	2275	2	3	134	77	211	165	13	84	13
22	9100	4	5	482	227	709	645	26	86	26
23	13860	2	3	473	182	655	688	40	86	24

Table B.3: Desharnais 2 data set

no	effort	team_exp	man_exp	transact	entities	p_a	p_na	enverg	yr_end	length
1	710	1	1	145	38	183	168	27	86	9
2	2429	4	4	174	78	252	267	41	87	9
3	651	2	2	126	49	175	180	38	88	3
4	1435	2	4	289	88	377	351	28	87	11
5	847	1	4	158	59	217	180	18	88	4
6	2352	2	4	661	132	793	698	23	87	34
7	546	0	4	97	42	139	99	6	86	6
8	595	0	2	213	73	286	203	6	84	6
9	1400	4	4	229	169	398	414	39	85	12
10	5880	4	3	469	176	645	697	43	86	12

Table B.4: Desharnais 3 data set

name	effort	duration	ksloc	fp	ua_fp
1	28700	17	253.6	1217.1	1010
2	8250	7	40.5	507.3	457
3	110731	15	450	2306.8	2284
4	8690	18	214.4	788.5	881
5	33630	13	449.9	1337.6	1583
6	8400	5	50	421.3	411
7	2320	5	43	99.9	97
8	13030	11	200	993	998
9	11600	14	289	1592.9	1554
10	7200	5	39	240	250
11	25870	13	254.2	1611	1603
12	23070	31	128.6	789	724
13	15700	20	161.4	690.9	705
14	24690	26	164.8	1347.5	1375
15	6990	14	60.2	1044.3	976

Table B.5: Kemerer data set

Bibliography

[AB04] M. Auer and S. Biffl. Increasing the accuracy and reliability of analogy-based cost estimation techniques with extensive project feature dimension weighting. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering (ISESE'2004) (accepted for publication)*, August 2004.

[ABRB05] M. Auer, C. Becker, A. Rauber, and S. Biffl. Implicit analogy-based cost estimation using textual use case similarities. In *The 3rd ACS/IEEE International Conferenceon Computer Systems and Applications (submitted)*, January 2005.

[AG83] A. Albrecht and S. Gaffney. Software function, source lines of code and development effort prediction: A software science validation. *IEEE Transactions of Software Engineering*, 9(6):639–648, 1983.

[AGB03a] M. Auer, B. Graser, and S. Biffl. An approach to visualizing empirical software project portfolio data using multidimensional scaling. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI'2003)*, October 2003.

[AGB03b] M. Auer, B. Graser, and S. Biffl. A survey on the fitness of commercial software metric tools for service in heterogeneous environments: Common pitfalls. In *Proceedings* *of 9th IEEE International Software Metrics Symposium (METRICS 2003)*, September 2003.

[AGB04] M. Auer, B. Graser, and S. Biffl. Visualizing software project analogies to support cost estimation. In *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS'04)*, April 2004.

[AKK93] A. Ali, M. Kalwani, and D. Kovenock. Selecting product development projects: Pioneering versus incremental innovation strategies. *Management Science*, 39:255–274, March 1993.

[AKY$^+$01] R. Agarwal, M. Kumar, M. Yogesh, S. Mallick, R. M. Bharadwaj, and D. Anantwar. Estimating software projects. *Software Engineering Notes*, 26(4):60–67, 2001.

[Aue02] M. Auer. Measuring the whole software process: A simple metric data exchange format and protocol. In *Proceedings of 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'2002)*, June 2002.

[BB92] C. Briand and V. Basili. A pattern recognition approach for software engineering data analysis. *IEEE Transactions on Software Engineering*, 18(11):931–942, 1992.

[BBHL94] B. Boehm, P. Bose, E. Horowitz, and M. Lee. Software requirements as negotiated win conditions. In *Proceedings of the 1st IEEE International Conference on Requirements Engineering (ICRE'94)*, pages 74–83, April 1994.

[BEB98] L. Briand, K. El Emam, and F. Bomarius. COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the 20th Inter-*

national Conference on Software Engineering (ICSE'98), pages 390–399, April 1998.

[Bec90] J. Becker. Entwurfs- und konstruktionsbegleitende kalkulation. *Kostenrechnungspraxis—Zeitschrift für Controlling (German)*, (6):353–358, 1990.

[BF85] W. Berry and S. Feldman. *Multiple Regression in Practice.* Sage Publication, 1985.

[BFOS84a] L. Breiman, J. Friedman, R. Ohlsen, and C. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[BFOS84b] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[BG96] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications.* Springer, 1996.

[BHM+00] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. Clark, B. Steece, A. Winsor Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II.* Prentice Hall, 2000.

[Bif04] S. Biffl. Collaborative interactive decision support. Technical report, Vienna University of Technology, 2004.

[BLW00] L. Briand, T. Langley, and I. Wieczorek. A replicated assessment and comparison of common software cost modeling techniques. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE'00)*, pages 4–11, Limerick, Ireland, June 2000.

[BM95] R. Bisio and F. Malabocchia. Cost estimation of software projects through case based reasoning. In *Proceedings of the 1st International Conference on Case Based Reasoning Research and Development, 1995*, pages 11–22, 1995.

[Bod98] J. Bode. Decision support with neural networks in the management of research and development: Concepts and application to cost estimation. *Information and Management*, 34(1):33–40, 1998.

[Boe81] B. Boehm. *Software Engineering Economics.* Prentice Hall, 1981.

[Boe90] B. Boehm. *Software Risk Management.* IEEE Press, 1990.

[Boe01] G. Boetticher. Using machine learning to predict project effort: Empirical case studies in data-starved domains. In *Proceedings of the Model Based Requirements Workshop*, pages 17–24, 2001.

[Bol97] T. Bollinger. The interplay of art and science in software. *IEEE Software*, pages 128, 125–126, October 1997.

[CC96] D. Clouse and G. Cottrell. Discrete multi-dimensional scaling. In G. Cottrell, editor, *Proceedings of the 18th Annual Conference of the Cognitive Science Society (COGSCI'96)*, pages 290–294, 1996.

[CD82] A. Coxon and P. Davies. *Key Texts in Multidimensional Scaling.* Heinemann, 1982.

[CDS86] S. Conte, H. Dunsmore, and V. Shen. *Software Engineering Metrics and Models.* Benjamin/Cummings, 1986.

[CI99] D. Cleland and L. Ireland. *Project Manager's Portable Handbook.* McGraw-Hill, 1999.

[CKSW97] N. Cassaigne, M. Kromker, M. Singh, and S. Wurst. Decision support for effective bidding in a competitive business environment. In *Proceedings of the 1997 IEEE International Conference on Computational Cybernetics and*

Simulation, volume 4, pages 3591–3596. IEEE, October 1997.

[CM96] D. Chatzoglou and L. Macaulay. A review of existing models for project planning and estimation and the need for a new approach. *International Journal of Project Management*, 14(3):174–183, 1996.

[Coc00] A. Cockburn. *Writing Effective Use Cases.* Addison-Wesley, 2000.

[Des89] J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, University of Montreal, 1989.

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

[EJ03] M. Engwall and A. Jerbrant. The resource allocation syndrome: The prime challenge of multi-project management? *International Journal of Project Management*, 21(6):403–409, May 2003.

[Eme71] J. Emery. Cost/benefit analysis of information systems. Technical Report 1, The Society for Information Management, 1971.

[Ern97] H. Ernst. The patent portfolio for strategic R&D planning. In *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET'97)*, pages 491–496, July 1997.

[ES03] H. Ernst and J. Soll. Integrating market and patent portfolios for market-oriented R&D planning. In *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET'03)*, pages 121–132, July 2003.

[FP96] N. Fenton and S. Pfleger. *Software Metrics - A Rigorous and Practical Approach.* PWS, 2nd edition, 1996.

[FW96] G. Finnie and G. Wittig. Ai tools for software development effort estimation. In *Proceedings of the International Conference on Software Engineering: Education and Practice(SE:EP'96)*, pages 346–353. IEEE Computer Society Press, January 1996.

[FW97] G. Finnie and G. Wittig. A comparison of software effort estimation techniques: Using function points with neural networks, case based reasoning and regression models. *Journal of Systems Software*, 39:281–289, 1997.

[Gib94] W. Gibbs. Software's chronic crisis. *Scientific American (International Edition)*, pages 72–81, September 1994.

[GM97] A. Gray and S. MacDonell. A comparison of techniques for developing predictive models of software metrics. *Information and Software Technology*, 39(6):425–437, 1997.

[GM99] A. Gray and S. MacDonell. Software metrics data analysis–exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, 4:297–316, 1999.

[GMS99] A. Gray, S. MacDonell, and M. Shepperd. Factors systematically associated with errors in subjective estimates of software development effort: The stability of expert judgment. In *Proceedings of the 6th IEEE International Software Metrics Symposium (METRICS'99)*, pages 216–229, November 1999.

[Gra92] R. Grady. *Practical Software Metrics for Project Management and Process Improvement.* Prentice Hall, 1992.

[GSW95] G. Goodhill, M. Simmen, and D. Willshaw. An evaluation of the use of multidimensional scaling for understanding brain connectivity. *Philosophical Transactions of the Royal Society*, B 348:265–280, 1995.

[GT97] J. Gordon and A. Tulip. Resource scheduling. *International Journal of Project Management*, 15(6):359–370, 1997.

[HC01] G. Heinemann and W. Councill. *Component Based Software Engineering: Putting the Pieces Together.* Addison-Wesley, 2001.

[HCYM98] R. Hughes, A. Cunliffe, and F. Young-Martos. Evaluating software development effort model-building techniques for application in a real-time telecommunications environment. *IEE Proceedings Software*, 145(1):29–33, 1998.

[Hee92] F. Heemstra. Software cost estimation. *Information and Software Technology*, 34(10):627–639, 1992.

[HHA91] J. Hihn and H. Habib-Agahi. Cost estimation of software intensive projects: A survey of current practices. In *Proceedings of the 13th International Conference on Software Engineering (ICSE'91)*, pages 276–287, May 1991.

[HS99] K. Heidenberger and C. Stummer. Research and development project selection and resource allocation: A review of quantitative modeling approaches. *International Journal of Management Reviews*, 1(2):197–224, June 1999.

[Hug96] R. Hughes. Expert judgement as an estimating method. *Information and Software Technology*, 38(2):67–75, 1996.

[HVK99] M. Hendriks, B. Voeten, and L. Kroep. Human resource allocation in a multi-project R&D environment. *Inter-*

national Journal of Project Management, 17(3):181–188, 1999.

[IKA02] A. Idri, T. Khoshgoftaar, and A. Abran. Can neural networks be easily interpreted in software cost estimation? In *Proceedings of the World Congress on Computational Intelligence (WCCI'02)*, pages 1162–1167, May 2002.

[Jon98] C. Jones. *Estimating Software Costs*. McGraw-Hill, 1998.

[Jør95] Magne Jørgensen. Experience with the accuracy of software maintenance task effort prediction models. *IEEE Transactions on Software Engineering*, 21(8):674–681, 1995.

[Jør04] Magne Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2):37–60, 2004.

[JRW00] R. Jeffery, M. Ruhe, and I. Wieczorek. A comparative study of two software cost modeling techniques using multi-organizational and company-specific data. *Information and Software Technology*, 42(14):1009–1016, 2000.

[Kee81] P. Keen. Information systems and organizational change. *Communications of the ACM*, 30(5):24–33, 1981.

[Kem87] C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, pages 416–429, May 1987.

[KI94] D. Kocaoglu and M. Iyigun. Strategic R&D program selection and resource allocation with a decision support system application. In *Proceedings of the 1994 IEEE International Conference on Management in Transition: Engineering a Changing World.*, pages 225–232. IEEE, October 1994.

[KKL+00] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self-organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, May 2000.

[KKMG90] D. Kira, M. Kusy, D. Murray, and B. Goranson. A specific decision support system (SDSS) to develop an optimal project portfolio mix under uncertainty. *IEEE Transactions on Engineering Management*, 37(3):213–221, August 1990.

[KL97] B. Kitchenham and S. Linkman. Estimates, uncertainty and risks. *IEEE Software*, 14(3):69–74, 1997.

[KN90] B. Kitchenham and B. Neumann. Cost modelling and estimation. In P. Rook, editor, *Software Reliability Handbook*, pages 333–376. Elsevier Applied Science, 1990.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[Koh95] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, 1995.

[KPLJ03] B. Kitchenham, L. Pickard, S. Linkman, and P. Jones. Modeling software bidding risks. *IEEE Transactions on Software Engineering*, 29(6):542–554, June 2003.

[Kru64a] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[Kru64b] J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

[Kru00] Philippe Kruchten. *The Rational Unified Process: An Introduction.* Addison-Wesley Longman, 2000.

[KT85] B. Kitchenham and N. Taylor. Software project development cost estimation. *The Journal of Systems and Software*, 5:267–287, 1985.

[KW78] J. Kruskal and M. Wish. *Multidimensional Scaling*. Sage, August 1978.

[LCB98] A. Lee, C. Cheng, and J. Balakrishnan. Software development cost estimation: Integrating neural networks with cluster analysis. *Information and Management*, 34(1):1–9, 1998.

[Lee01] J. De Leeuw. Multidimensional scaling. In *International Encyclopedia of the Social and Behavioral Sciences*. Elsevier, 2001.

[Lie98] E. Lieb. How many R&D projects to develop? *IEEE Transactions on Engineering Management*, 45(1):73–77, February 1998.

[LP96] A. Lederer and J. Prasad. A causal model for software cost estimation error. *IEEE Transactions on Software Engineering*, 24(2):137–148, 1996.

[LS87] J. Larkin and H. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[Luk00] C. Lukesch. *Umfassendes Projektportfoliomanagement in Dienstleistungskonzernen am Beispiel eines großen, international operierenden Versicherungsunternehmens*. PhD thesis, ETH Zürich, 2000.

[LWK$^+$00] J. Linton, S. Walsh, B. Kirchhoff, J. Morabito, and M. Merges. Selection of R&D projects in a portfolio. In *Engineering Management Society, Proceedings of the 2000 IEEE*, pages 506–511, 2000.

[LWM02] J. Linton, S. Walsh, and J. Morabito. Analysis, ranking and selection of r&d projects in a portfolio. *R&D Management*, 32(2):139–148, 2002.

[Mar98] M. Martin. En ERP strategy. *Fortune*, pages 95–97, February 1998.

[Max01] K. Maxwell. Collecting data for comparability: Benchmarking software development productivity. *IEEE Software*, pages 22–25, September/October 2001.

[MC00] E. Mendes and S. Counsell. Web development effort estimation using analogy. In *Proceedings of the 12th Australian Software Enginering Conference (ASWEC'2000)*, April 2000.

[MJ03] K. Moløkken and M. Jørgensen. A review of surveys on software effort estimation. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03)*, September/October 2003.

[MKL+00] Carolyn Mair, Gada Kadoda, Martin Lefley, Keith Phalp, Chris Schofield, Martin Shepperd, and Steve Webster. An investigation of machine learning based prediction systems. *The Journal of Systems and Software*, 53(1):23–29, 2000.

[MMC02] E. Mendes, N. Mosley, and S. Counsell. The application of case-based reasoning to early web project cost estimation. In *Proceedings of the 26th Annual International Software and Applications Conference (COMPSAC'02)*, August 2002.

[MS99] I. Myrtveit and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering*, 25(4):510–525, July/August 1999.

[MTN$^+$91] Y. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada. Method to estimate parameter values in software prediction models. *Information and Software Technology*, 33(3):239–243, 1991.

[MTON94] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki. Robust regression for developing software estimation models. *Journal of Systems and Software*, 27(1):3–16, 1994.

[MVP92] T. Mukhopadhyay, S. Vicinanza, and M. Prietula. Examining the feasibility of a casebased reasoning model for software effort estimation. *MIS Quarterly*, pages 155–171, 1992.

[MvW96] K. Maxwell and L. van Wassenhove. Software development productivity of european space, military and industrial applications. *IEEE Transactions on Software Engineering*, 22(10):706–718, 1996.

[NMB02] E. Nasr, L. McDermid, and G. Bernat. Eliciting and specifying requirements with use cases for embedded systems. In *Proceedings of the 7th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'02)*, pages 350–357, January 2002.

[Pfl98] S. Pfleeger. *Software Engineering: Theory and Practice.* Prentice Hall, 1998.

[PKLJ02] L. Pickard, B. Kitchenham, S. Linkman, and P. Jones. Evaluation of the software bidding model. Technical Report TR/SE 0204, Keele University, 2002.

[PS03] U. Passing and M. Shepperd. An experiment on software project size and effort estimation. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03)*, pages 120–131, September/October 2003.

[Put78] L. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, 4(4):345–361, July 1978.

[RGC00] J. Ringuest, S. Graves, and R. Case. Conditional stochastic dominance in R&D portfolio selection. *IEEE Transactions on Engineering Management*, 47(4):478–484, November 2000.

[RJW03] M. Ruhe, R. Jeffery, and I. Wieczorek. Cost estimation for web applications. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pages 285–294, May 2003.

[RL87] P. Rousseeuw and A. Leroy. *Robust Regression and Outlier Detection.* Wiley, 1987.

[RM99] A. Rauber and D. Merkl. The SOMLib digital library system. In S. Abiteboul and A.M. Vercoustre, editors, *Proceedings of the 3rd European Conference on Research and Advanced Technology for Digital Libraries (ECDL'99)*, number LNCS 1696 in Lecture Notes in Computer Science, pages 323–342, Paris, September 1999. Springer.

[RM03] A. Rauber and D. Merkl. Text mining in the SOMLib digital library system: The representation of topics and genres. *Applied Intelligence*, 18(3):271–293, 2003.

[RMD02] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, November 2002.

[RS02] N. Robinson and M. Shapcott. Data mining information visualisation beyond charts and graphs. In *Proceedings of the 6th International Conference on Information Visualisation (IV'02)*, pages 577–583, July 2002.

[Rub85] H. Rubin. A comparison of cost estimation tools. In *Proceedings of the 8th International Conference on Software Engineering (ICSE'85)*, pages 174–180. IEEE Press, August 1985.

[SA01] I. Stamelos and L. Angelis. Managing uncertainty in project portfolio cost estimation. *Information and Software Technology*, 43:759–768, 2001.

[Sal89] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley, 1989.

[SB93] G. Subramanian and S. Breslawski. Dimensionality reduction in software development effort estimation. *Journal of Systems and Software*, 21(2):187–196, 1993.

[SC01] M. Shackelford and D. Corne. Collaborative evolutionary multi-project resource scheduling. In *Proceedings of the Congress on Evolutionary Computation (CEC'01)*, volume 2, pages 1131–1138, May 2001.

[SED97] B. Samson, D. Ellison, and P. Dugard. Software cost estimation using an Albus perceptron (CMAC). *Information and Software Technology*, 39(1):55–60, 1997.

[Ser95] C. Serluca. An investigation into software effort estimation using a back propagation neural network. Master's thesis, Bournemouth University, 1995.

[SF95] K. Srinivasan and D. Fisher. Machine learning approaches to estimating development effort. *IEEE Transactions on Software Engineering*, 21(2):126–137, 1995.

[SH03] C. Stummer and K. Heidenberger. Interactive R&D portfolio analysis with project interdependencies and time profiles of multiple objectives. *IEEE Transactions on Engineering Management*, 50(2):175–183, May 2003.

[SJP02] F. Sheldon, K. Jerath, and O. Pilskalns. Case study: B2B e-commerce system specification and implementation employing use-case diagrams, digital signatures and XML. In *Proceedings of the 4th International Symposium on Multimedia Software Engineering (MSE'02)*, pages 106–113, December 2002.

[SM98] E. Stensrud and I. Myrtveit. Human performance estimating with analogy and regression models: An empirical validation. In *Proceedings of the 5th International Symposium on Software Metrics (METRICS'98)*, pages 205–213, November 1998.

[Som94] I. Sommerville. *Software Engineering.* Addison-Wesley, 1994.

[SS90] R. Staudte and S. Sheather. *Robust Estimation and Testing.* Wiley, 1990.

[SS97a] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–743, November 1997.

[SS97b] I. Sommerville and P. Sawyer. *Requirements Engineerings: A Good Practice Guide.* Wiley, 1997.

[SSK96] M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *Proceedings of 18th International Conference on Software Engineering (ICSE'96)*, pages 170–178, March 1996.

[SSS86] L. Schroeder, D. Sjoquist, and P. Stephan. *Regression Analysis: An Introductory Guide.* Sage, 1986.

[TG97] R. Tan and Q. Gong. A PC-based decision support system for r&d project selection. In *Proceedings of the Portland International Conference on Management of Engi-*

neering and Technology (PICMET'97), page 396, July 1997.

[The01] K. Thearling. Visualising data mining models. In *Information Visualisation in Data Mining and Knowledge Discovery*. Morgan Kaufman, July 2001.

[TM96] K. Torii and K. Matsumoto. Quantitative analytic approaches in software engineering. *Information and Software Technology*, 38(3):155–163, 1996.

[vG91a] M. van Genuchten. On the use of software cost models. *Information and Management*, 21:37–44, 1991.

[vG91b] M. van Genuchten. Why is software late? an empirical study of reasons for delay in software development. *IEEE Transactions on Software Engineering*, 17(6):582–590, 1991.

[VK94] M. Vigder and A. Kark. Software cost estimation and control. Technical Report NRC 37116, Institute for Information Technology, National Research Council Canada, Ottowa, Ontario, February 1994.

[VL88] A. Vepsalainen and G. Lauro. Analysis of R&D portfolio strategies for contract competition. *IEEE Transactions on Engineering Management*, 35(3):181–186, August 1988.

[VMP91] S. Vicinanza, T. Mukhopadhyay, and M. Prietula. Software-effort estimation: An exploratory study of expert performance. *Information Systems Research*, 2(4):243–262, 1991.

[VV95] R. Venkatraman and S. Venkatraman. R&D project selection and scheduling for organizations facing product obsolescence. *R&D Management*, 25(1):57–70, January 1995.

[Web96] B. F. Webster. The real software crisis. *Byte*, January 1996.

[Wie01] I. Wieczorek. *Improved Software Cost Estimation - A Robust and Interpretable Modeling Method and a Comprehensive Empirical Investigation.* PhD thesis, Fraunhofer Institut für experimentelles Software Engineering, 2001.

[WJ99a] F. Walkerden and R. Jeffery. An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 4:135–158, 1999.

[WJ99b] F. Walkerden and R. Jeffery. An empirical study of analogy-based software effort estimation. *Empirical Software Engineering*, 42:135–158, June 1999.

[WR02] I. Wieczorek and M. Ruhe. How valuable is company-specific data compared to multi-company data for software cost estimation? In *Proceedings of the 8th International Symposium on Software Metrics (METRICS'02)*, pages 237–248, June 2002.

[ZBGK01] W. Zuser, S. Biffl, T. Grechenig, and M. Köhle. *Software Development mit UML und dem Unified Process.* Pearson Studium, 2001.

Curriculum Vitae—Martin Auer

Personal Data

- Born on September 27, 1974 in Brunico, Italy
- Nationality: Italian

Education

- "Magister" in Mathematics, University of Vienna, Austria (passed with honors), 1993-00
- "Dipl-Ing." in Computer Science , Vienna University of Technology (passed with honors), 1993-99
- Exchange Student at the Universidad Politécnica de Cataluña, Barcelona, Spain (Erasmus fellowship), 1998
- Scientific high school, Brunico, Italy (passed with 60/60 points), 1988-93

Employment Positions Held

- Business Information Consultant at Raiffeisen Central Bank, Vienna (management information systems), 2002-
- Project Manager at RNG, Vienna (Web-based controlling systems), 2001
- Software Developer at T-Systems, Vienna (mobile health care systems), 2000
- Trainer at Austrian Chamber of Commerce's Institute of Business Promotion, (information technology), 1995-

Research and Teaching

- Lecturer at the Vienna University of Technology (undergraduate and graduate courses), 2002-
- Research Assistant at the Institute of Software Technology (mathematical models and metrics in software), 2002-
- Research Assistant at the Institute of Applied and Numerical Mathematics (high-performance computing), 1997-99

Miscellaneous Activities and Memberships

- Team Leader of the open-source project UMLet (graphical modeling of software systems), 2000-
- Founder of Auer Mayr IT OEG (IT consulting), 1999-
- Member of Assoc. for Computing Machinery (ACM); Inst. of Electrical and Electronical Engineers (IEEE)

Langages

- German (mother tongue); Italian (fluent); English (fluent); Spanish (good command)