TKK Dissertations 31
Espoo 2006

SOFTWARE DEVELOPMENT CONTROL PANELS

Concepts, a Toolset and Experiences

Doctoral Dissertation

Casper Lassenius

Helsinki University of Technology
Department of Computer Science and Engineering
Software Business and Engineering Institute

TKK Dissertations 31
Espoo 2006

SOFTWARE DEVELOPMENT CONTROL PANELS

Concepts, a Toolset and Experiences

Doctoral Dissertation

Casper Lassenius

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 22nd of May, 2006, at 12 noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Software Business and Engineering Institute

Teknillinen korkeakoulu
Tietotekniikan osasto
Ohjelmistoliiketoiminnan ja -tuotannon laboratorio

Distribution:
Helsinki University of Technology
Department of Computer Science and Engineering
Software Business and Engineering Institute
P.O. Box 9210
FI - 02015 TKK
FINLAND
URL: http://www.soberit.tkk.fi
Tel. +358-9-451 4851
Fax +358-9-451 4958
E-mail: reports@soberit.tkk.fi

ISBN 951-22-8196-1
ISBN 951-22-8197-X (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: http://lib.tkk.fi/Diss/2006/isbn951228197X/

TKK-DISS-2135

Otamedia Oy
Espoo 2006

HELSINKI UNIVERSITY OF TECHNOLOGY
P. O. BOX 1000, FI-02015 TKK
http://www.tkk.fi

ABSTRACT OF DOCTORAL DISSERTATION

Author Casper Lassenius

Name of the dissertation
Software Development Control Panels: Concepts, a Toolset and Experiences

Date of manuscript 23.4.2006 | Date of the dissertation 22.5.2006

[X] Monograph [ ] Article dissertation (summary + original articles)

Department	Department of Computer Science and Engineering
Laboratory	Software Business and Engineering Institute
Field of research	Software Engineering
Opponent(s)	Dr. Adele Goldberg, Neometron Inc.
Supervisor	Professor Reijo Sulonen
(Instructor)	Professor Reijo Sulonen

Abstract

This thesis discusses concepts and a toolset for as well as experiences with defining and implementing on-line visual displays of software development progress, *control panels*. Control panels group sets of measurement visualizations providing a multi-dimensional, integrated view of development status. Control panels are developed at various organizational levels, serving the information needs of different stakeholders. Navigation between different control panels enables users to drill down to view more detailed information on, e.g., a particular project or on the progress of a particular development activity.

In the constructive part of the work, I describe concepts, and a toolset that can be used for defining and implementing control panels. In addition, I provide guidelines for grouping measures into control panels, and discuss how measurement programs can be partitioned into control panels. The toolset, built as a "proof-of-concept" consists of a central measurement database, a visualization client, and a set of data importers that transfer data from external corporate repositories into the measurement database.

The empirical part discusses our efforts to define and implement control panels in four high-technology companies in Finland. Framed in the context of "constructive action research", the cases are first described individually, according to the main steps of the action research cycle, then compared in a cross-case analysis. Our results—we succeeded in implementing control panels in only one of four organizations—indicate that the development of control panels requires a substantial investment and commitment by the organization, but that it can be feasible even in small organizations. The main obstacles hindering the implementation were organizational and human in nature, more related to measurement program implementation in general than to tool support per se. Interestingly, the organizations did not consider measurement definition, i.e., the selection of "metrics" to be problematic. Success factors identified included management commitment, sufficient resourcing, and personnel involvement.

While limited, our experiences with the use of control panels show that they were felt to be more valuable by management than by the collocated project personnel. We hypothesise that the usefulness of control panels increases with project size and geographical distance. Thus, in the future it would be interesting to develop control panels for distributed software development projects, in which informal status communication is more difficult.

Keywords software measurement, software engineering, case studies, measurement tool support

ISBN (printed) 951-22-8196-1	ISSN (printed) 1795-2239
ISBN (pdf) 951-22-8197-X	ISSN (pdf) 1795-4584
ISBN (others)	Number of pages 148 p.

Publisher Software Business and Engineering Institute, TKK

Print distribution Software Business and Engineering Institute, TKK

[X] The dissertation can be read at http://lib.tkk.fi/Diss/2006/isbn951228197X/

TEKNISKA HÖGSKOLAN
PB 1000, FI-02015 TKK
http://www.tkk.fi

SAMMANFATTNING (ABSTRAKT)
AV DOKTORSAVHANDLING

Författare Casper Lassenius

Titel
Kontrollpaneler för programvaruutveckling: koncept, verktyg och erfarenheter

Inlämningsdatum för manuskript 23.4.2006 | Datum för disputation 22.5.2006

☒ Monografi ☐ Sammanläggningsavhandling (sammandrag + separata publikationer)

Avdelning	Avdelningen för databehandling
Laboratorium	Laboratoriet för programvaruaffärsverksamhet och -produktion
Forskningsområde	Mjukvaruproduktion
Opponent(er)	Dr. Adele Goldberg, Neometron Inc.
Övervakare	Professor Reijo Sulonen
(Handledare)	Professor Reijo Sulonen

Sammanfattning (Abstrakt)

Avhandlingen diskuterar koncept, datatekniska verktyg samt erfarenheter av definition och implementering av *kontrollpaneler* för programvaruutveckling. Kontrollpaneler grupperar visualiseringar av mätdata och kan sålunda erbjuda en flerdimensionell och integrerad insyn i programvaruvaruutvecklingens tillstånd. Kontrollpaneler utvecklas på olika organisationsnivåer för att på så sätt kunna tillfredsställa olika intressenters informationsbehov. Navigering mellan panelerna möjliggör detaljstudie, t.ex. av information om ett specifikt project eller tillståndet av en utvecklingsaktivitet.

Arbetets konstruktiva del beskriver en konceptuell ram och datatekniska verktyg som kan användas för att definiera och implementera kontrollpaneler. Den konceptuella ramen definierar grundläggande begrepp för definiering av kontrollpaneler samt presenterar principer för partitionering av panelerna. De datatekniska verktygen implementerade som *proof-of-concept*, innefattar en central mätdatabas, en visualiseringsklient samt ett antal importmoduler med vars hjälp mätdata kan importeras från externa databaser.

Den empiriska delen diskuterar våra erfarenheter av att definiera och implementera kontrollpaneler i fyra högteknologiföretag i Finland. Forskningsmetodiskt presenterat som konstruktiv aktionsforskning, fallstudierna beskrivs först individuellt, enligt aktionsforskningens huvudfaser, varefter de jämförs i en korsfallsanalys. Våra resultat—vi lyckades implementera kontrollpaneler endast i en av fyra organisationer—indikerar att utveckling av kontrollpaneler kräver en substantiell investering och organisationellt engagemang, men att det kan genomföras med framgång till och med i små organisationer. De värsta hindren för implementeringen var relaterade till organisationella och mänskliga faktorer, mera förknippade med implementation av mätprogram i allmänhet än specifikt med kontrollpanelimplementation eller verktygsstöd. Intressant var att organisationerna inte fann val av mätare svårt. Vi identifierade också framgångsfaktorer såsom ledningens engagemang, tillräcklig resursering samt personalens medverkan.

Våra erfarenheter visade att kontrollpanelerna upplevdes mera värdefulla av ledningen än av personalen som arbetade i samma kontor. Vi tror att kontrollpanelers utilitet ökar med project storlek och ökat geografiskt avstånd. Sålunda kunde det i framtiden vara intressant att utveckla paneler för distribuerade project, i vilka informell statuskommunikation är svårare.

Ämnesord (Nyckelord) programvaruutveckling, mätning, fallstudie, kontrollpanel, mätverktyg

ISBN (tryckt)	951-22-8196-1	ISSN (tryckt)	1795-2239
ISBN (pdf)	951-22-8197-X	ISSN (pdf)	1795-4584
ISBN (övriga)		Sidantal	148 s.

Utgivare Laboratoriet för programvaruaffärsverksamhet och -produktion, TKK

Distribution av tryckt avhandling Laboratoriet för programvaruaffärsverksamhet och -produktion, TKK

☒ Avhandlingen är tillgänglig på nätet: http://lib.tkk.fi/Diss/2006/isbn951228197X/

To Ulla^2 *(&) Maria*

Acknowledgements

> Unprovided with original learning, unformed in the habits of thinking, unskilled in the arts of composition, I resolved to write a book.
>
> Edward Gibbon

The practical work for this thesis was conducted in the Lucos research project at the TAI Research Institute at Helsinki University of Technology during the years 1997–2000. The members my team were instrumental in performing the work. Thank you Maarit Nissinen and Kristian Rautiainen for sharing your thoughts and ideas and the wonderful work you did in the company pilots. The measurement toolset was developed by four extremely competent individuals. Thank you Kai Risku, Jari Vanhanen, Pekka Kilponen and Jarno Vähäniitty for the outstanding work.

Without the interest and support of the case organizations, we would not have been able to get any empirical experiences in trying out our approach. I am deeply grateful for the time and effort you all put into the project, and for all comments, ideas and wisdom you conveyed to me and my research team during the three years we worked together.

Writing this dissertation has not been easy. Without the strong support and guidance of my teacher and good friend, professor Reijo "Shosta" Sulonen, this thesis would never have seen the lights of day. His belief in me and my abilities has been extremely valuable during the periods of self-doubt and intense self-critisism that I have gone through. The final write-up would probably never have taken place, had he not "strongly encouraged" me to do it. For that, I am eternally grateful.

Finally, without the loving support of my wife Maria, my parents, and the supporting attitude of a large number of people, too many to mention here, I would never have managed to finally get my thoughts sorted out and written down. Thank you all!

Hattula, April 2006
Casper Lassenius

Abstract

This thesis discusses a concepts, guidelines, a toolset and experiences with defining and implementing on-line visual displays of software development progress, *control panels*. Control panels group sets of measurement visualizations providing a multi-dimensional, integrated view of development status. Control panels are developed at various organizational levels, serving the information needs of different stakeholders. Navigation between different control panels enables users to drill down to view more detailed information on, e.g., a particular project or on the progress of a particular development activity.

In the constructive part of the work, I describe concepts, and a toolset that can be used for defining and implementing control panels. In addition, I provide guidelines for grouping measures into control panels, and discuss how measurement programs can be partitioned into control panels. The toolset, built as a "proof-of-concept" consists of a central measurement database, a visualization client, and a set of data importers that transfer data from external corporate repositories into the measurement database.

The empirical part discusses our efforts to define and implement control panels in four high-technology companies in Finland. Framed in the context of "constructive action research", the cases are first described individually, according to the main steps of the action research cycle, then compared in a cross-case analysis. Our results—we succeeded in implementing control panels in only one of four organizations—indicate that the development of control panels requires a substantial investment and commitment by the organization, but that it can be feasible even in small organizations. The main obstacles hindering the implementation were organizational and human in nature, and more related to measurement program implementation in general than to tool support per se. Interestingly, the organizations did not consider measurement definition, i.e., the selection of "metrics" to be problematic. Success factors identified included management commitment, dedicated measurement personnel, and organizational change management.

While limited, our experiences with the use of control panels show that they were felt to be more valuable by management than by the collocated project personnel. We thus hypothesise that the usefulness of control panels increases with project size and geographical distance. In the future it would be interesting to develop control panels for distributed software development projects, in which informal status communication is more difficult.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The beginning is the half of every action.

Greek Proverb

1.1 Background and Motivation

What is the state of our current software development project? Will we meet the goals regarding content, schedule, cost and quality? Is our software process improving? Will we complete the project on time? Is the quality of the software adequate? How much have the requirements changed compared to the original specification?

Questions like these are commonly asked by software practitioners, and being able to answer them using objective data whenever possible, is clearly preferable to guessing. To be able to do this, a company must define and install a *software measurement program* (e.g., Kitchenham, 1996; Fenton and Pfleeger, 1997).

1.1.1 Software Measurement

Measurement is a general tool that can be used to help us *understand* software development, to *monitor* status, to build models that help *predict* what will happen in development, to *control* development[1], and to *improve* our software development processes. (Briand et al., 1996)

In the field of software engineering, measurement has been an active area of research since the 1970s. Literally thousands of measures have been proposed, or at least so has been claimed (Fenton and Neil, 2000), case studies and experience reports testifying to the successful implementation of measurement programs in industry have been written (e.g., Daskalantonakis, 1992; Grady and Caswell, 1987; Grady, 1992)), and lots of advice on defining and setting up corporate measurement

[1]The term *control* is in this work understood in the sense of *managers using feedback to ensure that inputs, processes and outputs are aligned to achieve organizational goals* (Simons, 2000, p.52) as opposed to the use of the term in its statistical process control sense.

programs is available in the literature (e.g., Basili, 1984; Kitchenham, 1996; Pulford et al., 1996; Park et al., 1996; Florac et al., 1997)). Process improvement approaches such as the Capability Maturity Model (Humphrey, 1989; Paulk, 1995) and the Experience Factory (Basili and Rombach, 1988) also include measurement as a required component of high-quality software processes. In short, measurement can be considered a cornerstone of modern software engineering management, as evidenced, e.g., by its inclusion as a knowledge area in the software engineering body of knowledge (Abran et al., 2004).

1.1.2 Measurement Control Panels

In practical use of measurement for decision making and control, *visual displays*, such as graphs and charts are commonly used. It has even been proposed (Meyer, 1994; Brown, 1996) that management of development could be a "simple" as driving a car or flying an airplane if the person in the "driver's seat" had the same constant view of critical status information as provided by the *control panel* in cars and planes. Such a control panel for software engineering is shown in Figure 1.1.

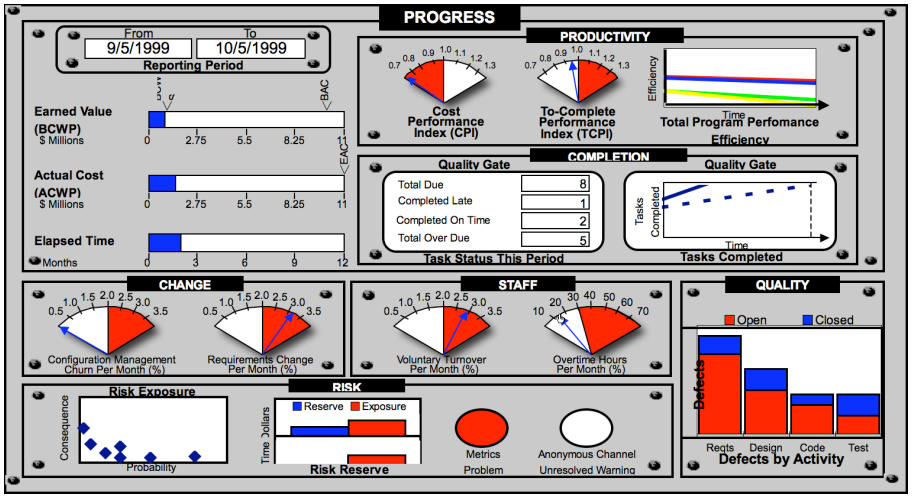


Figure 1.1: The Software Project Manager's Control Panel (Anonymous, 2000b)

While it is clear that managing software development is in many ways strikingly different from driving a car or an airplane[2], I believe that providing instant visual feedback on the status of software development is interesting, and might have several benefits, including:

- it may increase the commitment and motivation for practitioners regarding measurement,
- it may facilitate fact-based discussions and management of software engineering, and

[2]For example, I agree with Kitchenham (1996) that we are not dealing with simple parameters that are directly controllable.

- it may, combined with suitable mechanisms for measurement target setting, help an organization better achieve its goals

While the examples given by Meyer (1994) and Brown (1996) focus on a single control panel available at the team level, the idea easily lends itself to extension. In this work, I discuss our experiences in applying the control panel idea with three extensions: the definition of *role specific* visualizations, and subsequently the application of the idea on *different organizational levels* and to *different processes*; and finally, the ability to *link control panels together*.

The first extension is natural, since different stakeholders have varying needs for information. For example, senior managers need more aggregated information giving an overview of development status, while software practitioners need detailed information, e.g., regarding defects or modifications to the modules they are working on.

While having a control panel at the project level probably is the natural place to start, the idea can also be applied at other levels, from strategic measurement to portfolio and process monitoring and control. We thus propose to utilize the control panel idea at different organizational levels, and to different processes within the organization.

Having a set of control panels instead of a single one also leads to the question of linkages between the measures visualized in a panel and other panels, or between panels themselves. For example a "process panel" could show aggregate information on all projects executed according to a specific process, e.g., by showing an aggregate project plan (Kaplan and Norton, 1992). From this, one could open another, project-specific panel by clicking on a project in the aggregate view. Thus we can envision both linkages between specific measures visualized in a panel and another panels, and between panels themselves. An example of the second case could be a project panel containing a button to a "Test phase" panel.

We thus envision a software development organization using a set of interlinked control panels developed according to the needs of various stakeholders. This vision is depicted in Figure 1.2.

1.1.3 The Lucos Research Project

The work discussed in this thesis was carried out between September 1997 and March 2000 in a research project called *Lucos* and performed jointly by the TAI Research Center at Helsinki University of Technology and five participating companies. In the Lucos project, we aimed at improving the companies' ability to control their product development efforts.

The work we did with the companies included defining and modeling their development strategies and processes. In that work, our overarching goal was to develop control panels and the underlying measurement systems and tool support that they required. In this thesis, I focus on our efforts to develop and implement control panels, and as such it represents but a small part of the actitves we did in the research project.

Figure 1.2: An Example Control Panel Hierarchy

Having discussed the motivation and background of the research, we are now ready to turn to the objectives and research questions of this thesis.

1.2 Research Goal and Objectives

The main objective of the research is:

> *To explore the design, implementation and use of control panels for software development in industrial settings.*

To do this, we basically need to tackle three issues and their related questions:

1. **The design of control panels:** what panels and for whom should they be developed? What measures should be included in the panels? How should the measures be visualized? How should the panels be linked together?

2. **The "foundations":**. Visualizing the status and goals of an organization is only the "tip of the iceberg". Getting to the point where control panels can be implemented requires that the organization has measurable goals with defined targets, and a working measurement system that captures the data in a form suitable for use in control panels. In this work, when I talk about

"foundations", I refer to this large area of all organizational practices and processes that make the measurement data needed in the control panels available.

In organizations without clearly defined processes and working measurement systems, building the foundations for control panels can be a major undertaking. Fortunately, the literature can provide lots of help in process and measurement system development. Thus, in this work, I merely try to identify the necessary foundation, and provide examples of approaches from the literature that can be used when working to build the needed foundation.

3. **Tool Support:** While certainly possible to implement most of the ideas using manual labor, in practice it by necessity requires automation. Thus, the final issue that must be dealt with is tool support both for the foundation and for the control panels themselves.

To tackle these issues, I break down the problem into the following three sub-objectives:

1. to develop a conceptual framework and guidelines for the development of control panels for software development
2. to develop a toolset supporting control panel development and deployment in order to enable testing the ideas in practice
3. to test the concepts, guidelines and toolset in industrial environments

Each objective is further described and motivated below.

1.2.1 Development of Conceptual Framework and Guidelines

While the development of control panels is intimately related to the development and maintenance of the corporate software measurement program, it is useful to view them as separate processes that can be integrated in various ways. For example, it is possible to build control panels on top of an existing measurement system, without defining new measures; as is an approach in which the control panel development drives the development of the measurement system itself.

There are, as reviewed in Chapter 2, several approaches available for the development of measures and measurement systems. However, the systematic development of visualizations and their collection into control panels is typically not included in these. Since our focus is on the development of control panels, we are not interested in proposing yet another approach for measurement program development, but instead focus on the development of the visualizations and their combination into control panels. To this end, this work develops a framework for control panel development that identifies the main concepts needed. In addition, I try to provide tentative guidelines that can help practitioners when defining control panels. The requirements for this are listed in Table 4.1 and further discussed in Chapter 4.

Table 1.1: Requirements for conceptual framework and guideline development

No	Requirement
FR-1	*Concepts:* The framework should define the central concepts needed for the development of control panels.
FR-2	*Guidance:* Guidelines for control panel development should be developed.
FR-3	*Method independence:* Control panel development should be independent of measurement paradigm or methodology.
FR-4	*Organizational independence:* It should be possible to develop control panels for different organizations and environments.
FR-5	*Technological independence:* Control panel development should be independent of the particular technological tools deployed.
FR-6	*Usability:* The framework should be easy to learn and use.
FR-7	*Feasibility:* Control panel development should be feasible in practice.

As discussed in Chapter 4, to satisfy these requirements, we developed a conceptual framework defining the main concepts needed for the development of control panels, as well as developed guidelines for the development of panels.

1.2.2 Toolset development

For measurement to be effective, especially when aiming at instant feedback of measurement results, automation is crucial (Grady and Caswell, 1987; Tian et al., 1997; Basili and Rombach, 1988; Lavazza, 2000; Parviainen et al., 1997; Kempkens et al., 2000). We derived a set of requirements for a toolset capable of supporting control panel development and deployment. The requirements are summarized in Table 1.2. At that point in time, we did not find any existing tools that contained the functionality needed, and we therefore set out to build our own tool for supporting the definition and display of measurement data. Later, tools such as MetricCenter and its successor DataDrill (Distributive, 2005) by Distributive Management[3] and recently HackyStat (Johnson et al., 2005) have implemented similar ideas. These, as well as the tools available at the time of our work are further discussed in Scetion 2.5

The toolset we developed to satisfy these requirements, as well as the rationale behind both the requirements and the toolset are further discussed in Chapter 5.

1.2.3 Empirical Testing

The final objective was to test the our concepts, guidelines and toolset in industrial environments. In practice, the research did not follow a strict linear "research-then-transfer" (Potts, 1993) model in which we first developed the concepts, guidelines and toolset and then tested them empirically. Instead, we worked

[3] `http://www.distributive.com`

Table 1.2: Requirements for toolset development

No	Requirement
TR-1	*Stakeholder customizability:* The system should support the definition and display of different, customizable views for different stakeholders.
TR-2	*Navigation:* The system should support navigation between different panels.
TR-3	*Access control:* The system should support authentication and access control.
TR-4	*Data source compatibility:* The system should be compatible with existing corporate measurement systems and data sources.
TR-5	*Multi-method approach:* The system should support different approaches to the definition and application of measurement.
TR-6	*Ease of use:* Basic system usage should be easy and not require special skills or training.
TR-7	*Multi-platform compatibility:* The system should work in both Windows and Unix environments, and must support both the Netscape and Internet Explorer browsers.
TR-8	*Distributedness:* The system should support people working in physically separate locations.
TR-9	*Low-cost:* The system must be cheap to acquire and should not contain any costly third-party components.
TR-10	*Logging:* The system should support logging of user actions.

closely together with our industrial partners for a period of three years, during which we gained both feedback and ideas for improvement. Working this way ensured the practical relevance of the work, as well as allowed the researchers to gain a deeper understanding of the problems and issues of measurement development in industry than possible if we had adopted a linear research model. The research approach used is described fully in Chapter 3.

1.3 Scope

The focus of this work is on the *development and implementation* of *control panels* for software development. Thus, the main focus is on the design and implementation of the control panels themselves.

In a perfect world, companies would have working measurement systems to which we could utilize an experimental research design (Campbell and Stanley, 1963), essentially "plugging in" our ideas and tools and then compare the post-treatment results with the pre-treatment ones. Unfortunately, as explained in Chapter 6, only one of the case companies had a working measurement system. Therefore, we will, on several occasions, have reason to touch upon general issues of measurement and measurement system design, i.e., the "foundations", though a thorough discussion of them is outside the scope of this work.

Furthermore, since the focus is on software development, the literature review does not cover other fields, such as manufacturing or management that routinely use measurement for status information and control. The one exception to this is a short discussion on the Balanced Scorecard (BSC) (Kaplan and Norton, 1992), which several of our partners either were using, or were adopting. While strictly viewed not in the focus of the work, the BSC is briefly discussed in the literature review chapter in order to make it easier for the reader to follow the discussions on our empirical experiences in Chapter 6.

The validation efforts in this work must be considered very tentative, since the companies also participated in and influenced the design of the conceptual framework, the guidelines and the toolset. Likewise, as discussed in Chapter 3, this kind of research also has a high risk for researcher bias. Thus, conclusions regarding the superiority or inferiority of our proposed approach as compared to other ones are not possible based upon the data of this study, but left for future research. However, most research in software engineering proposing new frameworks and approaches typically suffer from the same weaknesses.

1.4 Assumptions

In our work, we made the following assumptions. They are stated here to communicate the measurement "philosophy" taken in this work, not as a complete list.

Assumption 1 *Management decision making should be based upon facts rather than opinion whenever possible and feasible.*

The first assumption states that we think that management should be based upon hard facts rather than opinion also in software development, despite the problems related to getting this information. This *fact-based* view of management, as opposed to intuitive or informal management is the prevalent view in the field of software engineering.

Our second assumption is related to the application of measurement to organizations:

Assumption 2 *Software measurement can be successfully implemented in organizations.*

In addition to success stories, empirical evidence on the application of measurement to organizations also contain warning examples of measurement dysfunction leading to problems like gamesmanship and invalid data. In this work we take the stance that measurement dysfunction is not an inherent property of organizational measurement, but an accidental property resulting from, e.g., defining meaningless measures or misuse of measurement data.

The third assumption deals with human behavior:

Assumption 3 *Software developers and development managers have a positive and constructive attitude to their work and strive to perform well even under suboptimal circumstances.*

In his book on performance measurement, Simons (2000, p. 13) makes five points about humans working in organizations in "modern economies". First, they *want to contribute* to an organization of which they can be proud. Second, they know the difference between right and wrong, and *choose to do right.* Third, people *strive to achieve* even in the absence of external inducements, such as rewards and bonuses. Fourth, people *like to innovate*, and fifth, they *want to do competent work* and take pride in their accomplishments. Though, as admitted by Simons, somewhat heroic, these assumptions well capture the positive attitude to software developers taken in this work.

1.5 Contributions

This thesis makes three distinct contributions to knowledge. First, the conceptual framework and guidelines for control panel development presented in Chapter 4 presents a way of conceptualizing and structuring the presentation of measurement results in software development organizations. While the focus in this research has been on software development, the concepts and development process is likely to be applicable also to other fields, like new product development.

Second, the toolset discussed in Chapter 5 provides a platform that can be used both for developing and supporting control panels, as well as for analysing their use, thus deepening our understanding of benefits and drawbacks. While now somewhat technologically outdated, the toolset's value is probably more in its set of features and design than in the actual implementation.

Finally, our empirical experiences, discussed in Chapter 6, show that implemeting control panels is viable, but challenging. In particular, tool deployment and integration into existing measurement programs and tools is difficult and demands a high level of organizational commitment and a substantial investment. We identified tentative success factors and issues that must be dealt with when trying to implement control panels.

1.6 Thesis Structure

This initial chapter has discussed the motivation, goals and objectives, scope and central terminology used in this research, as well as the research approach used and its central contributions. The rest of the thesis is structured in the following way:

Chapter 2 contains review of the literature on software measurement focusing on the main issues from the point of view of developing tool supported software development control panels.

Chapter 3 presents the research design employed.

Chapter 4 discusses the conceptual framework and guidelines that we developed and used.

Chapter 5 presents a toolset we built to support the design and implementation of software development control panels.

Chapter 6 discusses our empirical experiences in applying the framework, guidelines and toolset in industry.

Chapter 7 contains the conclusions, discusses the contributions of the research, and outlines directions for future work.

Chapter 2

Theoretical Background

> When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: It may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the stage of science.
>
> Lord Kelvin, 1824–1904

2.1 Introduction

In this chapter I present an overview of software measurement literature relevant to the development and implementation of software development control panels. Since the existing body of knowledge on software measurement is vast[1], I focus on three aspects: approaches to development of industrial measurement programs, the use and development of visualizations and control panels, and measurement tool support. Before this, I present the central terms used. Since general overviews and historic descriptions of the software measurement field are available elsewhere (Fenton and Neil, 2000; Zuse, 1995, e.g.), I have not included such discussions here, but refer the interested reader to the above references.

2.2 A Note on Measurement Terminology

This section presents the key concepts and central terminology used in the thesis. This is necessary, since the terminology still is confusing and authors even use the same word in completely different meanings, making it necessary to infer the meaning from the context[2].

[1] A search in INSPEC on the term "software metric" gives over 5000 hits.

[2] As an example, the term *metric* has been used to refer to both the *instrument* used for measurement, as well as the *result* of that measurement, and its *presentation.* Thus the term "metrics reuse" could mean

2.2.1 Measurement

I have adopted the definition of *measurement* coined by Fenton and Pfleeger (1997, p. 5): *the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.* The resulting entities are referred to as *measures.*[3]

The important point about this definition is that when defining measures, we need to explicitly identify the *objects of measurement* in the real world, as well as their *attributes* that we intend to measure, and define the *rules* for assigning these values. The measurement rules are typically documented in a *measurement plan.* These rules, are important, e.g., for the purpose of interpreting the results of measurement. In organizational measurement, these rules need to define both the *raw data* to collect, as well as how to *transform* this data into *measurement values*

2.2.2 Visualizations and Control Panels

For the purpose of this work, the term *control panel* refers to *"a collection of measure visualizations or textual displays of software measurement data grouped into a two-dimensional display". Measure visualizations* refer to the *"visual display of software measurement data in the form of a chart, graph or other non-textual display". Textual displays* include the use of text, numbers and tables to show measurement data.[4]

2.2.3 Measurement Processes

Organizational measurement can best be understood as a collection of measurement related processes. I understand measurement as consisting of three distinct activities: *data collection*, *processing*, and *presentation*, as shown in Figure 2.1.

Data collection consists of collecting and storing raw data from the organization. Even though it could be philosophically argued that this in itself constitutes measurement, we have found it useful to define this as a separate activity when dealing with organizational measurement. The processing phase (sometimes referred to as analysis) consists of converting the raw data into measurement values according to defined rules, and presentation consists of presenting the measurement values by using charts, graphs and tables.

---

that the same measurement instrument should be applied in a different context, that the measurement results could be reused, perhaps for another purpose than originally envisioned, or that similar charts of graphs could be drawn in another context, even though, e.g., the details of data collection or analysis had changed.

[3] I have chosen to use the term *measure* instead of the often widely used (and poorly defined) term *metrics*, since I have found no compelling reason to adopt the term metric. Making the same decision, Park et al. (1996) states *"The problem with "metric" is not that no one knows what it means, but that everyone thinks it means something different. Measurement, on the other hand, has a generally accepted definition".*

[4] Initially, we developed control panels without textual displays. However, our practical experiences showed that users of control panels often were interested not only in the graphical view of a measure, but also in knowing the specific numeric result of a particular measure. Thus, I later expanded the definition of control panels to include textual display. In our toolset, for example, these were implemented both by allowing text fields, and by tooltips that show a particular measurement value when the cursor "hovers" over, e.g., a particular bar of a bar chart.

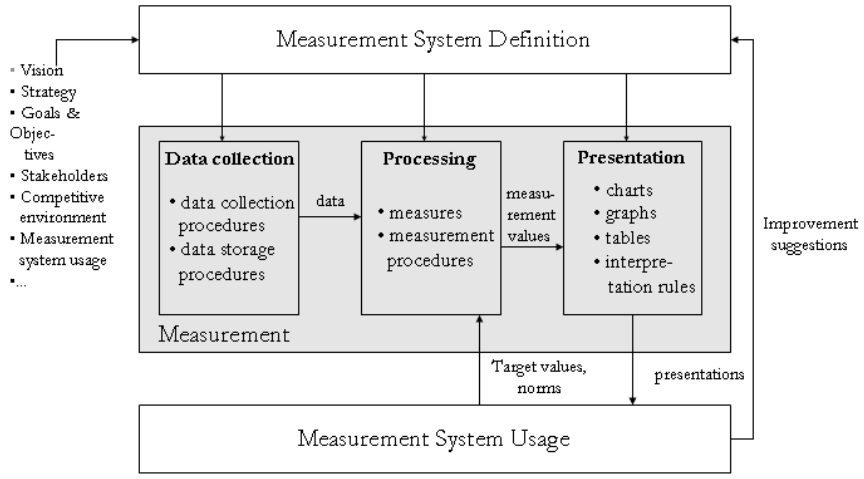


Figure 2.1: Measurement Related Processes

The figure also shows other measurement related processes. Here, an important distinction is made between the process that *defines* the measures and procedures to be used for measurement, and the actual process of carrying them out, the *measurement process*. Some authors, e.g. Fenton and Pfleeger (1997) consider both of these to be part of the same overall measurement process. This is, however, confusing in the context of organizational measurement, as the processes have different instantiation characteristics. Finally, the measurement system usage process is the interface to the measurement process that defines the target values (plans) for the various measures, as well as uses the presentations produced by the measurement.

From the point of view of developing and using control panels, their definition should be integrated into the measurement system definition process. The display of control panels is done in the presentation activity of the measurement process. As discussed later, the measurement process is a good candidate for automation, in particular the presentation activity.

2.2.4 Controlling and Monitoring

Measurement is a general mechanism that can be used for various purposes, such as understanding, prediction, control and motivation (Briand et al., 1996; Kerssens-van Drongelen, 1999). In this work, I focus on the use of measurement for the purpose of *control* and *monitoring*.

Control can be defined as *the process of comparing actual performance with planned performance, analyzing variances, evaluating possible alternatives, and taking appropriate corrective action as needed* (Anonymous, 2000a). Thus, when measuring for the purpose of control, it is crucial that we have a *plan* that indicates the intended *target values* of our measures. The plan typically includes a temporal aspect, i.e. a statement of what value the measure should behave at a certain point in time. This connection to a plan distinguishes this kind of measurement from *monitoring*, in which case we only are interested in following up some attribute, but lack the intention of making it meet a predefined goal.

Control panels can contain visualizations both of measures used for control, in which case the target values are typically shown in the same visualization as the measurement results (actuals), and of measures used for monitoring, in which case the target values are not shown.

Next, we will turn our attention to the development of control panels and their constituent visualizations, as discussed in the software engineering literature.

2.3 Measurement Program Development

While lots of individual measures have been proposed[5], it is clear that for practical purposes any single measure is not enough, but a set of measures is needed. Thus, a company wanting to measure its software development effort needs to develop a *measurement program* that addresses what measures to collect, and how to collect and report them. We will next look at available approaches to do this.

During the 80s and 90s, several approaches for the development and implementation of software measurement in industry were developed. The approaches can basically be grouped into three philosophical schools: the *fixed-set of measures school*, the *goal-based measurement school*, and the *statistical process control school.*

2.3.1 Fixed Sets of Measures

The first school takes the basic point of view that there is a set of measures that can be developed, validated, and subsequently applied everywhere. Being the most traditional approach, it has led to the software engineering community being very active in developing a large number of measures for various purposes. Fenton and Neil (2000) states that the literature contains thousands of proposed measures, most of which have not been validated and have never been used in practice. While the statement might be intentionally provocative and exaggerated, there is no doubt that the number of proposed measures is very large, and the use of many of them outside the environment in which they were developed have not been reported.

Proponents of this approach includes consultants, such as Putnam and Myers (1997) and the Airlie council, a set of software engineering experts that developed a set of project management measures for application to any and all projects (Brown,

[5] E.g., Fenton and Neil (2000) claims that thousands of measures have been proposed in the field of software engineering.

1996; Anonymous, 1998b). Table 2.1 shows the measures included in the Arlie council project control panel. Carefully constructed, the panel contains measures grouped into five groups: progress, change, staff, risk, and quality, and provide a balanced view of the state of a project. Early reports of measurement success in industry, in particular (Grady and Caswell, 1987) might be classified as belonging to this group; however later reports from the same authors clearly belong to the goal-based measurement school (Grady, 1992).

Despite having many benefits, and widely used in other fields, in particular quality management of manufacturing, this approach is problematic when applied to software development since it assumes that all development efforts are similar and thus can use the same measures. While all software development undetakings share characteristics, such as the reliance upon a skilled workforce and a creative component, I strongly believe that factors such as the type of software, team characteristics, process characteristics, and organizational characteristics all affect the set of measures that should be applied. For example, I find it unlikely that a five-person project developing a simple Web application using, e.g., php would need and use the same measures as a 300 person project developing real-time critical software for a telecommunications switching system.

However, there are many clear benefits from having a predefined, fixed set of measures. For example, building tool support is much simpler, and it does make it possible to compare and benchmark. It is also beneficial from an organizational point of view, since measures do not have to be redeveloped for each new development effort, but instead relies on heavy measure reuse. Taking the SPMN control panel as an example of this point, deploying it as is over a large number of organizations and projects is in many ways much simpler than any approach or toolset that requires both the definition of new measures and supporting such arbitrary measures. After successful deployment, projects might be easily benchmarked using a panel such as the one by SPMN[6].

Thus, while probably an impossible goal to achieve at the general level, I think it is worthwhile to strive for a set of standardized set of measures that can be reused, with qualification, within the same organization. Then the crucial question becomes at what level measures and sets of measures can be reused.

2.3.2 Goal-based Measurement

Despite the striking benefits of having a fixed set of measures that could be applied to all situations, the second approach, goal-based measurement, has been much more successful in practice. Goal-based measurement relies upon the idea that measures must be tied to the organizational context, in particular to organizational goals, and that there is no set of measures that can be standardized and universally applied.

The academically most influential work developing industrial measurement programs was performed by Basili and his colleagues at the University of Maryland by developing the Goal-Question-Metric (GQM) approach (Basili and Rombach,

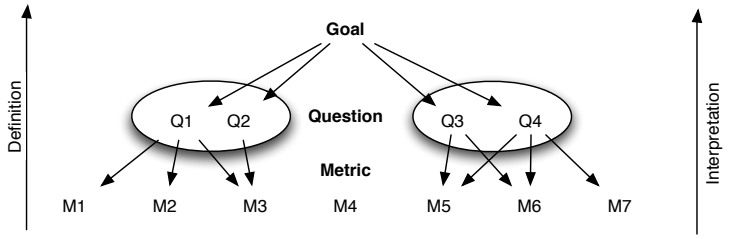
[6]Given that the users are able to correctly interpret the measures.

Table 2.1: Measures in the SPMN Project Control Panel(Anonymous, 1998b)

Group	Measure	Explanation /Comments
Progress	Earned value	Budgeted cost of work performed (BCWP) compared with total budgeted cost (BAC) and budgeted cost of work performed (BCWS)
	Actual cost	Actual cost of work performed (ACWP) compared with total estimated cost (EAC)
	Elapsed time	End of current reporting time period vs. total budget (SAC)
	Cost performance index (CPI)	BCWP/ACWP
	To-complete performance index	$\frac{\text{BAC}-\text{BCWP}}{\text{EAC}-\text{ACWP}}$
	Total program performance efficiency	Abba chart
	Quality gate task status	Number of tasks due, completed on time, completed late, and total overdue tasks last month
	Quality gate tasks completed	Cumulative number of tasks planned and completed over time
Change	Configuration management churn per month	$\frac{\#\text{ of modified CI's rechecked into CM last month}}{\#\text{ of CI's in CM system}} \cdot 100$
	Requirements change per month	$\frac{\#\text{ of new and changed requirements last month}}{\#\text{ of original requirements}} \cdot 100$
Staff	Voluntary turnover per month	$\frac{\#\text{ of staff voluntarily leaving last month}}{\#\text{ of staff at beginning of month}} \cdot 100$
	Overtime per month	$\frac{\#\text{ overtime hours last month}}{\#\text{ of base hours}} \cdot 100$
Risk	Risk exposure	Each risk plotted as a dot
	Risk reserve	Cost: risk exposure cost vs. risk reserve cost. Time: schedule risk reserve vs. schedule risk cost
	Metrics problem	Metrics warning indicator
	Anonymous warning	Bad news from staff
Quality	Defects by activity	Critical and severe defects open and closed

1988; Basili et al., 1994). This approach, which is grounded in the authors' experience with software measurement at the NASA Software Engineering Laboratory[7], is based upon the idea that measures must be developed based upon an organization's explicitly stated measurement goals. The authors' experiences with collecting measurement data (Basili, 1984), showed that collecting data without a clear view of what it will be used for is likely to be counterproductive and lead both to missing data and the collection of data that will never be used, but stored in "data graveyards".

Using the GQM, measures are defined top-down by first stating formal *measurement goals*, then asking questions that need to be answered in order to meet the goal, and finally, define measures that answer the questions. This exercise helps build a GQM tree that makes implicit the mental models people have of the object under study. This knowledge is important when interpreting the results of the measures. Figure 2.2 shows the goal tree, and a goal template used by the GQM approach. While not originally part of the GQM approach, the goal template was developed to help formulate measurement goals. As shown in the figure, each measure can be tied to several questions, and several questions to each goal.



(a) A GQM Tree (Basili, 1984)

Dimension	Description
Purpose	To *(characterize, evaluate, predict, motivate etc.)* the (*process, product, mode, metric, etc.)* in order to *(understand, assess, manage, engineer, learn, improve, etc.)* it
Perspective	Examine the *(cost, effectiveness, correctness, defects, change, product metrics, reliability, etc.)* from the point of view of the *(developer, manager, customer, corporate perspective, etc.)*
Environment	The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints etc.

(b) The GQM Goal Template (Basili and Rombach, 1988)

Figure 2.2: The GQM Approach

While intuitively nice and simple, the GQM has not been without its critics. More specifically, it has been deemed to be more an approach for brainstorming than a real method (e.g. (Card, 1993)), due in particular to its lack of a guiding

[7]Card (1993) mentions that it is a generalization of the Factor/Criteria/Metric framework developed at Rome Laboratories and reported by McCall (1977).

process and the gap between the definition of goals and the selection of measures. Partly addressing this problem, subsequent work on GQM has developed extensions to the approach, such as the abstraction sheet (Briand et al., 1996), as well as documented the GQM process in greater detail (Solingen and Berghout, 1999).

The GQM approach has also become popular in industry – many of the existing experience reports, like (Grady, 1992; Daskalantonakis, 1992; Fuggetta et al., 1998) have based their measurement programs on the GQM, albeit the earlier ones quite informally.

In addition to the work expanding upon the GQM, the most well-known approaches to building industrial measurement programs, including the **ami** approach (Pulford et al., 1996), and the GQ(I)M approach (Park et al., 1996) are goal-driven. Common to these approaches is the identification of *business goals* that are further broken down into GQM *measurement goals*, thus adding another layer of abstraction to the process.

Goal-based measurement is also widely adopted in fields outside software measurement. Many *performance measurement* approaches rely on the definition of goals and deriving related measures. In particular the popular Balanced Scorecard (BSC) approach for strategic measurement (Kaplan and Norton, 1992) belongs to this group. The BSC requires companies to define measurable objectives in at least four perspectives: the *customer, internal business process, learning and growth,* and *financial* perspectives. This is claimed to help management get a "balanced" view of the organization replacing purely financial management with other complementing perspectives[8]. The BSC and GQM can also be naturally combined, as proposed both by us (Lassenius et al., 1999) and others (Goethert and Fisher, 2003, e.g.).

The main problem with goal-based approaches like the GQM and its relatives lies in the idea of context-based measurement itself. Blindly subscribing to the idea of always developing new measurements for each new undertaking might, while perhaps given the right expertise and infinite resources, produce the best possible set of measures for each situation. However, this can lead, in addition to high measurement development costs, to challenges with benchmarking, tool support, and training the personnel. In practice, however, goals and related measures can often be reused, and new measures do not have to be developed from scratch each time.

2.3.3 Statistical Process Control

One of the key factors in increasing productivity in manufacturing has been the introduction of statistical process control (SPC), in particular the work of Juran, Shewart and Deming. While SPC is intended for controlling highly repetitive processes, typically of short duration, which means that one can quickly get a large number of runs, its application to software development, or any development

[8]While the BSC basically is outside the scope of this work since it belongs to the field of performance management, it is included here since many organizations at the time of our research had implemented, or were in the process of implementing scorecards.

activity, is not without problems. Development efforts in general, and software development in particular, is typically more or less one-of-a-kind and thus inherently non-repetitive, has long runs, and uses processes that hardly are in statistical control.

This has not put the proponents of SPC down. Instead, they claim, the non-repetive nature of software development at, e.g., the project level, does not mean that, e.g., certain subprocesses, such as testing or inspections are not highly repetitive. And, the goal, of course is to make software development as a whole in statistical control, using the classic quality tools.

The work on SPC for software development by Florac et al. (1997); Florac and Carleton (1999) is, in addition to laying out the SPC ideas, also philosophically related to the goal-based measurement school, since it contains the idea of developing organizationally specific measures that are explicitly related to the business goals of the organization. Since the main point, however, is to apply SPC, I have classified it as belonging to the SPC schoool. Other work on statistical process control in software engineering include Burr and Owen (1996) and Weller (2000).

Despite their best efforts, it seems that statistical process control in software engineering, and indeed the discipline of software engineering itself still have a long way to go before SPC can become mainstream practice. Despite this, the quality tools and visualizations used by the SPC community might have a lot to offer to software engineering, and can be useful even if statistical process control is not achieved. In particular, the basic charts and graphs used in quality management are interesting and mostly applicable to software development.

2.3.4 Summary

Summarizing the above discussion, we can conclude that there is a large body of knowledge on software measurement in general, and that there are several approaches available for the development of industrial software measurement programs. The existing approaches can be classified into three groups: those based upon defining and applying a fixed set of measures cross organizations and projects, those based upon deriving measures based upon organizational goals, and those based upon statistical process control. Of these, the goal-based school is undoubtedly the most popular. Table 2.2 lists the main measurement approaches developed in the field and their main ideas.

From the point of view of visualizing software development status, all of the reviewed schools can be useful.

2.4 Measurement Visualization and Control Panels

In this section, I discuss the literature on software visualizations. First, I look at the use and development of single measurement visualizations, followed by a discussion on how these can be combined into control panels.

Table 2.2: Measurement Development Approaches

Approach	Main ideas	Example references
SPMN	• Use a fixed set of measures and related "best practices" at the project level • Make development visible using a "control panel"	Brown (1996); Anonymous (1998b)
Pfleeger	• Measurement must be tied to process maturity. • Different measures can be introduced at different maturity levels	Pfleeger and McGowan (1990)
GQM and its extensions	• Measures must be derived based upon clearly stated measurement goals • Three level conceptual hierarchy: Goals, Questions, and Metrics • All levels are needed both for measure development and interpretation • Arrange feedback sessions with developers	Basili (1984); Basili and Rombach (1988); Fuggetta et al. (1998); Solingen and Berghout (1999, 2001)
ami	• Four activities: assess, analyze, metricate, improve • Uses business (primary) goals to derive measurement goals, from which questions and measures are derived • 12-step method • Developed by European consortium of academia and industry	Debou et al. (1994); Pulford et al. (1996)
M^3P (Model-Measure-Manage Paradigm)	• Recognizes stakeholders at different corporate levels • Includes success factors obtained by studying software measurement programs • Uses risk as a killing factor to avoid explosive growth of goals, subgoals, questions and measures	Offen and Jeffery (1997)
GQ(I)M	• Extends the GQM by introducing business goals at a higher level of abstraction than measurement goals • Adds the notion of *indicator* that is developed prior to the actual measure	Park et al. (1996); Goethert and Hayes (2001)
Statistical Process Control	• Based upon the ideas of Shewart and Deming • Apply basic SPC tools to software development processes • Goal is statistical control using control charts	Florac et al. (1997); Florac and Carleton (1999); Burr and Owen (1996)

2.4.1 Single visualizations

The idea of visualizing the state of software development using visual displays is not new. Most papers on software measurement, from the early days on, include example charts or graphs depicting the visualization of proposed measures. Books on software measurement, as well as published case-studies also routinely illustrate the measures used by showing charts and graphs. (e.g. Gilb (1977); Fenton and Pfleeger (1997)).

Visualizing data has several benefits, as opposed to using statistical analysis. Software engineering data is, e.g., typically not normally distributed, the data sets are small, and might have been collected in a manner that does not meet the requirements for experimental setups. Applying statistical tests randomly to such data might show spurious correlations. On the other hand, graphs are robust, because they do not require any statistical assumptions, and are more user friendly than statistical analyses. The visualizations must still be made with care, and the interpretation carried out carefully (Pulford et al., 1996; Park et al., 1996).

The fact that most software measurement articles contain example graphs means that literature contains a large set of proposed visualizations, some used only by the developers of a particular measure, others widely used. Despite this abundance of visualizations, I have not been able to find any systematic discussion on their use, or their "goodness" in the software engineering field. This is unfortunate, since one way of attacking the development of visualizations is to have a collection of "good" ones from which one could pick the ones that are appropriate, or at the least use as a basis for further customization. An informal study of case articles and software measurement books shows that most visualizations are based upon simple graphs of the kind that a modern spreadsheet program can draw. In addition, more exotic visualizations have been proposed, but do not seem to have become widely used (Curtis and Scarfone, 1992). The field of software visualization has developed ways of visualizing software artifacts, but these do not seem to be used in typical software measurement programs. However, there is literature on development of visualizations in general. In particular the work by Tufte (2001, 1997) might be helpful when developing software development control panels.

Of the reviewed measurement program development approaches, only GQ(I)M and **ami** included guidelines for developing visualizations. In the discussion on indicators in the GQ(I)M method, Park et al. (1996) state that *"As you can see, constructing useful indicators is a highly creative process. Unfortunately, it is often easy to construct nifty indicators that mislead both the creators and their audiences".*

From our point of view, the work at the Software Engineering Institute (SEI) is particularly interesting, since it is the only one of the reviewed approaches that includes the development of visualizations (called *indicators*) in the design phase of the measurement program. The approach, named GQ(I)M (Park et al., 1996) is, as the name suggests, based upon the GQM approach, with the addition of an extra level—the *indicator* level—between the questions and the measures. To aid in developing indicators, the GQ(I)M method includes an indicator template, the use of which has later been discussed by Park et al. (1996); Goethert and Hayes (2001); Goethert and Siviy (2004). According to this report, the indicator

template has been used successfully both to develop new measurement programs, as well as to focus and improve existing ones.

While undoubtedly useful, the indicator template does not provide any actual guidance for the selection of visualization type, i.e., type of graph for a certain measure. Instead, it focuses on motivating the need for the indicator, as well as on documenting the necessary measurement program details, such as what data to collect, defining data collection and analysis responsibilities and reporting. Thus, the template is more a general template for visualization-driven measurement program development than actually an aid when developing the visualizations. The indicator template is shown in Figure 2.3 below. Indicating the authors' practical experience, template customization is encouraged.

Figure 2.3: The GQ(I)M indicator template (Goethert and Hayes, 2001; Goethert and Siviy, 2004).

More practical advice is given in the **ami** method (Pulford et al., 1996), step 10 of which is "Presenting and distributing the data". The description of this step contains a list of graphical presentations, as well as some simple guidelines for developing graphs. These guidelines discuss the use of simple data plots, such as X-t plots, pie charts, scatter plots, and histograms, and give examples of data sets and measures to which these are applicable.

Practically useful is also the work on SPC. Florac and Carleton (1999), as well as other books pushing the use of statistical process control in software engineering (e.g. (Burr and Owen, 1996)) discuss the use of traditional SPC tools, including control charts.

2.4.2 Control Panels

For a complex activity like software development it is unlikely that any single measure could be developed that would contain enough information for various stakeholders. Instead, measurement programs consist of sets of measures. It is well understood that focusing on any single measure and excluding everything else can have unexpected and unwanted effects. Thus, the literature typically stresses the need to have a balanced set of measures to get a good picture of what is going on. For example, Park et al. (1996, p. 59) approaches this through the concept of unexpected data, stressing the importance of understanding a problem from various viewpoints, and in particular the importance of looking at other variables too, when seeing something "strange". If using the GQM approach, a single goal and question typically leads to several measures, that together help answer the question.

From the point of view of presenting the results of measurement, getting such a balanced view requires sets of visualizations. As discussed previously, these can be combined into a compound display, a control panel. The perhaps most widely published control panel is the one developed for software projects by the Software Project Manager's Network (SPMN). This panel contains a predefined set of measures, described in the *Program Manager's Guide to Software Acquisition Best Practices* (Anonymous, 1998b), and the context of which is discussed in (Brown, 1996). The visual appearance of the panel is shown in Figure 2.4.

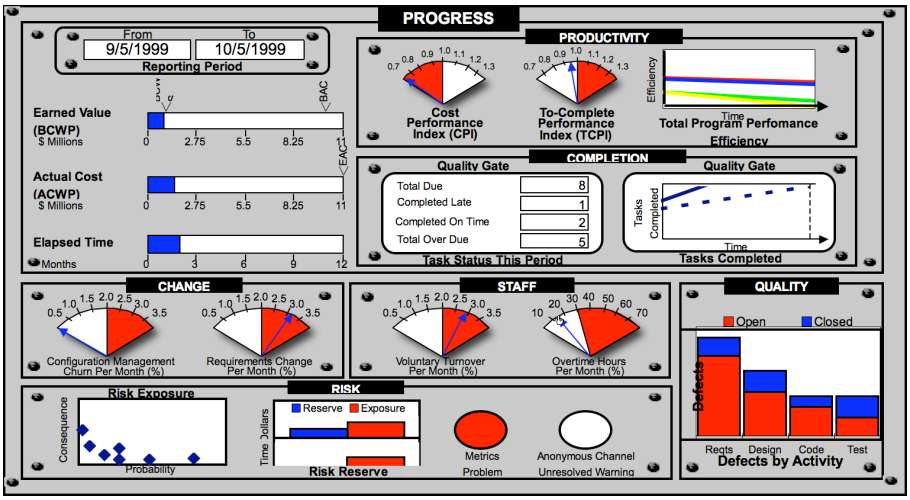


Figure 2.4: The Software Project Manager's Control Panel (Anonymous, 2000b)

As shown in the figure, the control panel displays project status in five dimensions: progress, change, staff, risk, and quality. Most dimensions contain several measures, giving a good overview of project status. While clearly dependent upon a particular process and reporting cycle, the measures in the panel seem well conceived, and in particular the dimensions are likely to be applicable to

most software development efforts. The control panel has been implemented as a freely available Excel application[9].

While an intuitively nice idea, the control panel concept is not without its critics. In particular, Kitchenham (1996, pp. 1–4) discusses both the underlying ideas of the control panel, as well as the specific measures, and identifies two main problems that she thinks are rooted in the "control panel" analogy:

- The visual presentation using speedometer dials and odometers gives the impression that all indicators are simple objective measures, and
- The implication that the measures are under control

The first problem is particularly evident when looking at measures based upon estimates, which in software engineering tend to be both subjective and poor. Evidence suggests that the best early estimates (done, e.g., in the requirements phase) are off by up to 100%, and that estimates made later in the process still may miss the target by as much as 30%. Thus, e.g., earned value tracking which is based upon original estimates might be grossly misleading.

Another problematic area related to the first issue, according to Kitchenham, is the use of *quality gates*. These are (preferrably objective) conditions that must be met before a task is considered complete. Kitchenham identifies two problems with such gates: the fact that a task might turn out not to have been completed after all, e.g., due to deficient execution; and that important tasks might be missing from the original plan. These problems can lead to the gauge winding both backwards and forwards, and even to unexpectedly change its value. While this observation most certainly is true, I am not sure that it poses such a problem—if we understand the issues involved, i.e., that it is not a linear or straight trip to the goal.

The problem of the assumption of the measures being under control is that, Kitchenham claims, it means that "good project managers, like good drivers" are expected to know what to do when a measure changes. However, what actions to take, and under what conditions to take them is not clear, and the control panel provides no guidance in this respect.

While there certainly is some truth in the critique, I don't totally agree. First, I don't think at all that simply having a nice visual layout resembling a control panel implies that the measures are neither objective nor in control. However, I agree that this is implied in the conceptual whole as presented by the report. However, since people in software engineering organizations typically are well aware of the problems in controlling software development, I don't think this threat is that big.

A more serious problem, I think, is the the idea that the same panel can be applied to all projects and organizations. While the SPMN report and Airlie council do not recognize the need to customize the control panel, the user manual for version 2.0 of the application shows that practical use seems to raise the need

[9]The application can be downloaded from the homepage of the Software Program Managers Network at `http://www.spmn.com/`.

for customization: *"Based on requests from users for more adaptable control panels that can be modified to reflect individual project metrics, Project Control Panel provides the option of creating additional custom control panel worksheets. These worksheets can be tailored to display metrics appropriate for your project that are not represented on the main control panel.*(Anonymous, 2000b). However, the viewpoint taken is that the measures included on the main control panel are applicable to and relevant for all projects.
.

Later work has developed more general frameworks for control panels as well as discussed problems of developing customized control centers. (Münch and Heidrich, 2004). In particular, this work proposes the use of roles as the basis for control panel development, and notes the lack of existing methodological support for eliciting information requirements from the roles. While I agree with the idea of defining role-based control panels, I think that, e.g., the GQ(I)M might be useful from the point of view of eliciting role visualization requirements, thus at least partly refuting the lack of methodological support. However, to my knowledge the literature does not contain any discussion on how to meaningfully combine single visualizations into control panels, either based upon stakeholder roles, or any other criteria. We will return to this issue in Chapter 4.

2.5 Tool Support

While it is possible to manually perform all necessary tasks when developing and drawing control panels, this is hardly feasible in practice. To make control panels implementation feasible in practice, typically both the control panels themselves, and their underlying measurement foundation requires tool support. In this section, I give a short presentation of related work.

A general classification and discussion of measurement related tools is out of the scope of this discussion. The interested reader is referred to the work by Dumke (1996); Dumke and Grigoleit (1997); Dumke and Winkler (1997). The focus here is on tools that can provide measurement analysis and presentation in the form of control panels.

In a recent article, looking at tool support for the development of role-based and purpose-oriented software project control centers, Münch and Heidrich (2004) reviewed seven academic environments and found support lacking—the tools included in their review and the main results is shown in Figure 2.5. The review classified tools into categories based upon the support for data presentation and visualization, as well as their support for data processing. On the data presentation axis, the tools are classified into those providing only a static set of visualizations, those that have the possibility to vary the visualization based upon intended usage purpose (e.g., monitoring, prediction), and those that in addition support role-based visualizations. On the data processing axis, the tools are classified into those providing support only for predefined functions, those adding context-oriented adaptation (e.g. for a particular project), and those providing variable functions and context-oriented adaptation. As can be seen in the figure, none of the reviewed environments supported role-based

visualizations using variable functions. The toolset that we developed, and which is discussed in Chapter 5 could, with some qualifications, be put in the uppermost right square, the main problem being that it per se does not support data analysis, but relies on other tools to provide data in a format suitable for visualization.

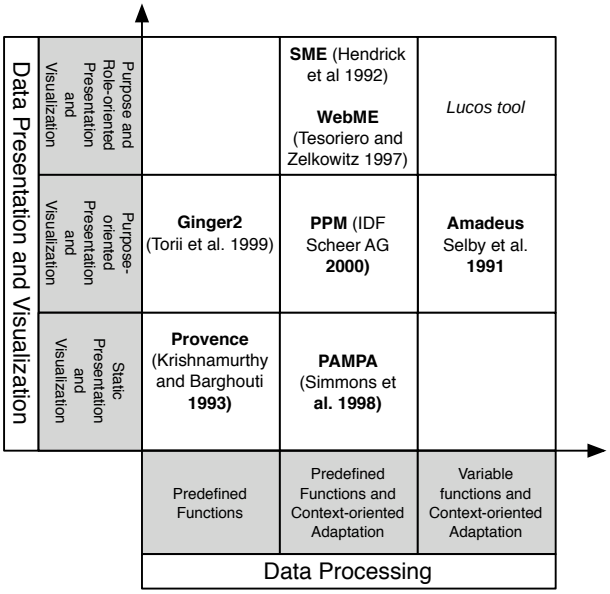


Figure 2.5: Tools Reviewed by Münch and Heidrich (2004)

Deciding not to repeat the same review here, I have omitted the tools reviewed in that article, and have added related tools that (Münch and Heidrich, 2004) did not include in their review. These tools and the reported experiences with their use is listed in Table 2.3.

The table lists three tools that were developed at the same time as our own research, and two recent developments. The software project control management consortium's project panel was one of the main motivations for our work, and was implemented as an Excel application. The initial version of the control panel did not support tailoring of project measures of their timing, but later versions added this, upon request from users. From the point of view of our goal, the SPMN control panel can be seen as one possible instantiation of a control panel for the project manager role. Unfortunately I have not been able to find any reported experiences with the use of the tool.

MetriFlame (Parviainen et al., 1997) is a tool providing support for the whole GQM measurement process, including measurement definition, data collection, analysis and presentation. The reported case study shows preliminary feasibility of adopting and using the tool, and more importantly, the need to tailor the presentations to the needs of various stakeholders.

The GQM Tool (Lavazza, 1998, 2000) is another tool supporting GQM-based measurement. The case study showed benefits such as increased acceptance of

Table 2.3: Related Measurement Tools and Experiences

Tool	Measurement School	Features and Benefits	Reported Experiences
Software Project Manager's Control Panel (Brown, 1996; Anonymous, 1998b, 2000b)	Fixed set of measures	• Simple Excel application with fixed measures • Requires manual data entry • Later versions added measure customizability	• ?!
MetriFlame (Parviainen et al., 1997)	Goal-based measurement	• Support for metrics definition, collection and result presentation • Connections to different data sources • Tailoring of presentations to the needs of different stakeholders	• Case study at ABB showed feasibility of approach, cost-effectiveness for feedback sessions, and need to tailor reports to specific stakeholder needs
GQM Tool (Lavazza, 1998, 2000)	Goal-based measurement	• Eased management of GQM plans (reuse, ensuring structural consistency, keeping track of large number of goals) • Linking GQM plan to corporate database (Access) • Data analysis (running queries) and display • Link to external measurement tools	• Case study at Pirelli Cavi showed low cost of automated data collection, prompt goal verification and evaluation, faster and cheaper GQM feedback sessions, and increased acceptance of metrics.
MetricCenter & DataDrill Dashboard (Distributive, 2005)	Paradigm independent?	• Definition of dashboards at several organizational levels • Role-based dashboards • Support for measurement data collection, analysis and visualization • Implements alerts to help focus attention	• Several companies listed as customers, including Booz-Allen Hamilton, Accenture, BAE Systems • Some press-releases published containing success stories. Very little useful information included.
HackyStat (Johnson et al., 2005)	Fixed set of measures	• Integrated into Eclipse platform • Automatic data collection • Simple analysis aiming at in-process monitoring and short-term prediction • Data drill-down supported	• Used with success on own development project • No industrial experiences as of yet

measurement, lower data collection and feedback costs, and reuse possibilities for measurement plans.

The commercial tool DataDrill dashboard provides multiple-level drill-down capabilities, hierarchical dashboards that can be customized, secure role-based visualizations and data access, and relies upon a web-based architecture. It indeed looks very similar to the tool we developed, as discussed in Chapter 5. However, it also provides some additional features, e.g., alarms that help focus management attention to important events or data. I was unfortunately not able to find any case studies reporting on the use of the tool. The press releases provided by the company which describe success stories, as well as the number of reference clients listed can be interpreted to show the feasibility of the approach.

The most recent tool based upon similar ideas is probably HackyStat (Johnson et al., 2005). HackyStat is integrated into the Eclipse development environment[10], and supports automatic data collection and simple analysis. The tool is extensible, i.e., it is possible to add new data collection modules and graphs, and aims at project-level support for in-process monitoring and short-term prediction. The tool has not yet been used in industrial environments, though the experiences with the development team using it for their own project has provided promising outlooks. Since the addition of new measures requires coding new data collection and presentation modules, I have classified HackyStat as belonging to the fixed set of measures school.

In 1997 when we started our work, we did not find any tool that met our requirements, and therefore we decided to develop our own tool. In addition, one of our pilot companies had performed a study of existing measurement systems on the market, and considered our approach superior to existing commercial systems. This hypothesis is further strengthened by the fact that other contemporary research projects developed tools in a similar vain, and that the review by Münch and Heidrich (2004) did not find tool support for role-based measurement.

Summarizing the discussion on tool support for control panels, we can see that several tools are available, but none (with the exception of DataDrill Dashboard, and possibly HackyStat) that explictly supported the definition and implementation of role-based, customizable control panels. However, the published case studies help us establish both the need for tool support, as well as the expected benefits of it.

2.6 Conclusions

This chapter reviewed the software measurement literature from the point of view of building a set of linked control panels visualizing the state of software development in an organization. The review was divided into three parts, each of which identified gaps in the literature.

The review on measurement program development methodologies concluded that there are a number of approaches available for developing industrial software measurement programs. The approaches, which were classified into three

[10] `www.eclipse.org`

schools all have something to contribute to the development of control panels, but they typically lack support for developing the actual visualizations, as well as for grouping them in a meaningful way into control panels. Finally, the section on tool support found that there are several frameworks and tools avilable for supporting software measurement. However, the reviewed tools did not support customizable control panels, with the exception of one commercial tool which has been developed subsequent to our efforts. The literature does, however, support the use of measurement, measurement visualizations, their grouping into control panels, and the need to support the control panels with suitable tools.

Thus, while detailed information on how to build the control panels is not available, the idea should be sound, at least according to the literature. We will next discuss the research design we deployed for trying out the control panel idea in our partner companies.

Chapter 3

Research Design

> It is not worthwhile to go round the world to count the cats in Zanzibar.
>
> Henry David Thoreau

3.1 Introduction

This chapter discusses the research design employed in this thesis. First,some general observations on research in software engineering are presented, Second, the approaches of *design research* and *action research* are presented, followed by the presentation of the approach used in this thesis, called "constructive action research" . The methodological evaluation is not presented here, but, in the interest of reader convenience, in Section 7.4 after the case study write-ups.

3.2 Research Methodology in Software Engineering

Software engineering is a young, immature science, which has even been characterized as being in a crisis (Glass, 1994). Researchers have attacked their own field in various ways: it has been described as "unscientific" (Fenton et al., 1994), not fulfilling the criteria for an engineering science (Shaw, 1990), being an "advocacy science" (Glass, 1994), relying too little on empirical validation of proposed ideas and constructs (Curtis, 1980; Basili et al., 1986; Tichy et al., 1995; Tichy, 1998; Glass et al., 2002), using an extremely narrow set of research methodologies (Glass et al., 2002), and having little or no impact on how software actually is developed in real-world organizations (Potts, 1993; Glass, 1994; Glass et al., 2002).

Critisizing is, however, easy. Fortunately several authors have also proposed solutions for improving the maturity of the field. These include:

- to use an *industry-as-laboratory* mode of research instead of the prevalent *research-then-transfer* mode, i.e., for researchers to work in close co-operation with industrial practitioners. This helps ensure research relevance, and can

help improve the impact of software engineering research. (Potts, 1993; Shaw, 1990; Glass, 1994)

- to focus on empirical validation of proposed ideas, concepts, models and methodologies (Curtis, 1980; Basili et al., 1986; Tichy et al., 1995; Tichy, 1998; Basili et al., 1986). This helps overcome the problem of "advocacy science", provided that the empirical validation is adequately performed.

- to learn and apply research methodologies and approaches from other fields, such as medicine and sociology. Kitchenham has particularly focussed on using analogies from research in medicine. (Kitchenham, 2004; Kitchenham et al., 2002).

Some researchers, perhaps most notably Victor Basili (Basili, 1996; Basili et al., 1999, 1986) have advocated applying the scientific paradigm used in the natural sciences to software engineering, pushing for more empirical research and experimentation. This seems to have been at least partly successful: there is now a more of less established community around "empirical software engineering", and there is at least one academic journal focusing on the subject. However, a study of software engineering journals over the years 1996–2001 showed that still only about 20% of published papers included empirical validation of their results. The majority of the papers were, not unexpectedly, in the category "design research". This is along the same lines as earlier studies, such as (Tichy et al., 1995; Zelkowitz and Wallace, 1998). Thus, looking at papers published in software engineering, the field is clearly constructive in nature, with empirical validation of constructs becoming increasingly common.

However, the knowledge on how to validate constructs, and how to do empirical studies in software engineering is still quite scarce. The field still lacks clear guidelines and approaches, though more work is constantly done in this respect (Wohlin et al., 2000, e.g.). Researchers have both classified different approaches to empirical studies (Zelkowitz and Wallace, 1998), and provided guidelines for doing empirical research (Kitchenham et al., 2002; Seaman, 1999; Juristo and Moreno, 2001). Much of this work has its roots in other fields, such as sociology or medicine. From this, we can learn that the appropriate strategy for construct testing is contingent on many things, e.g., the nature of the solution and its intended application domain (Zelkowitz and Wallace, 1998). In the product innovation literature, testing of constructs has been likened to Beta-testing, commonly performed in the software industry (Dolan and Matthews, 1993; Chiesa et al., 1996). In practice, the innovation and testing phases tend to overlap, as feedback from early testing can and should improve the product, i.e., the construct the researcher is developing.

In empirical research one typically strives for settings in which the researcher is *detached* from the object of study, ideally taking the role of an outside observer, intentionally keeping the interaction between the researcher and the object under scrutiny at a minimum, with zero being the goal. In software engineering research, in particular if the researcher has developed a new construct that needs to be

tested, this is often not possible. In practice, the researcher often needs to take the construct into an organization for testing, thus being heavily involved. Thus, the risk of researcher bias is significant. In addition, organizations that sign up for early studies might be classified as "early adopters" according to the model by Moore (1991), and consequently are more positive towards the approach than future mainstream organizations. Thus, early results can easily be overly positive.

3.3 Design Research

Good constructive, or design, research, requires more than knowing how to validate a construct. One framework that can be helpful, though not published in the field of software engineering, but in the information systems field, is the one by Hevner et al. (2004). Since IS and SE are related fields (Glass, 1992), I think that the framework can be applied quite straightforwardly to software engineering. The framework, shown in Figure 3.1, shows the two environments of design research: the *environment* (on the left), needed to ensure the practical relevance of the work, and the *knowledge base*, containing theoretical foundations and methodologies that the design research can use and should contribute to. Design research in itself is viewed as consisting of two processes: *build*, and *justify/evaluate*. It also shows the iterative nature of most design research: the artifacts are improved based upon the feedback from the justify/evaluate process.

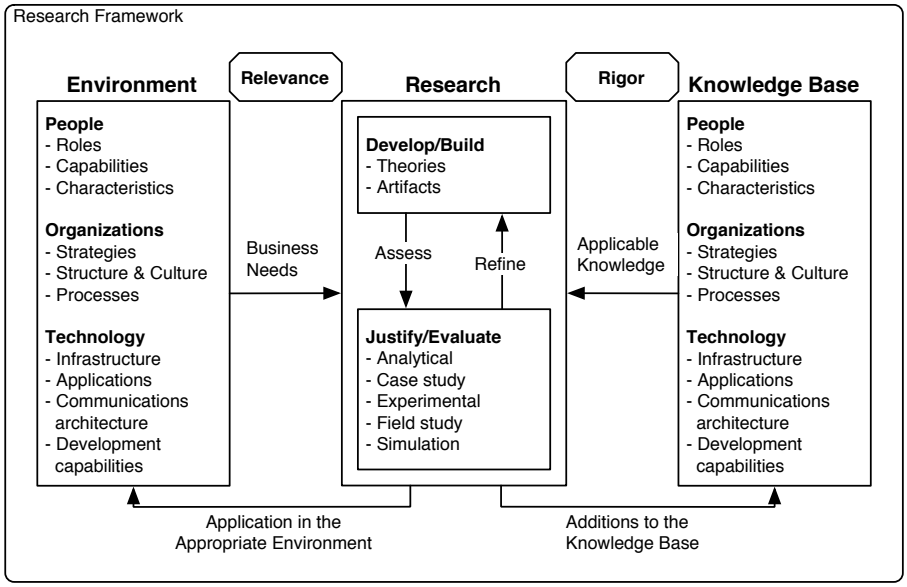


Figure 3.1: Design Research Framework (Hevner et al., 2004)

While Hevner et al. (2004) don't present any specific research process for doing design research, the six steps of *constructive research*, a term used in a similar

meaning, presented by Kasanen et al. (1991, 1993) provide a useful outline for our purposes. Constructive research is presented as consisting of six steps:

1. Searching for a problem which is both practically relevant and interesting from a research point of view
2. Gaining a preliminary understanding of the research problem
3. Innovation phase, in which the solution is created
4. Practical testing of the solution, i.e., verification of the construct
5. Exposition of the theoretical underpinnings of the solution and statement of the contribution to scientific knowledge
6. Examination of the scope of applicability of the solution

As explained in Section 3.5 below, these steps well describe the approach taken in this work. However, as indicated in the framework by Hevner et al. (2004) construction and testing were intimately interconnected, and done in close cooperation with our corporate partners during our research. A way of framing this is through the lens of *action research*, which is discussed in the next section. Before that, however, we shall take a short look at the guidelines for design research that Hevner et al. (2004) present, since they are useful for later evaluating the research. These guidelines are shown in Table 3.1 below. Since the guidelines are mostly self-explanatory, I refer the reader to the original text for a more in-depth discussion of them. The guidelines are also discussed in more detail in Section 7.4 where I evaluate the research of this thesis.

3.4 Action Research as a Validation Approach

Since we worked in a small number of companies, using the classifications shown in the framework discussed above, we used a *case study* approach to validation. (Yin, 1994, p. 13) defines case study as *"an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident"*. Such an approach is appropriate when the researcher is unable to control behavioral events, and the focus is on contemporary events. Furthermore, Yin (1994) states that the case study inquiry is suitable when there are more variables of interest than data points, and therefore relies on multiple sources of evidence and triangulation, and that case study research benefits from the prior development of theoretical propositions to guide data collection and analysis.

This characterization well fits a situation in which one wishes to test a new construct. When, as in our case, testing of the construct demands that the test environment (i.e., company) adopts a new way of working, a case study approach is appropriate. Action research can employ various methodologies, both qualitative and quantitative. In this research, the case studies are, as described earlier, best framed in the context of action research.

Table 3.1: Guidelines for Design-Science Research (Hevner et al., 2004).

	Guideline	Explanation
1.	Design as an Artifact	Design science must produce a viable artifact in the form of a construct, a model, a method, or an instantiation
2.	Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3.	Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
4.	Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
5.	Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
6.	Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7.	Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Action research is a qualitative research method that has its roots in the work of Kurt Lewin, and aims at bridging the gap between science and the practical affairs of the world (Stringer, 1999). Simultaneously and independently developed in England by a group working with World War II victims, action research *combines theory and practice (and researchers and practitioners) through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework* (Avison et al., 1999).

Action research has three distinct characteristics (Baskerville and Wood-Harper, 1996):

- The researcher is *actively involved*, with expected benefit for both *researcher* and *organization.*
- The *knowledge obtained* can be *immediately applied.* There is not the sense of the detached observer, but that of an active participant wishing to utilize any new knowledge based on an explicit, clear conceptual framework.
- The research is a cyclical process *linking theory and practice.*

There are, however, challenges with such an approach. Baskerville and Wood-Harper (1996) mention the lack of impartiality of the researcher and that the process easily leads to a lack of research discipline. Gummesson (2000) notes that action research sometimes is critisized as "consulting masquerading as research". Overcoming these problems, requires—in addition to lobbying research

colleagues—the establishment of a formal research agreement, provision of a theoretical problem statement, using planned measurement, to maintain collaboration and subject learning, to promote iterations and to be careful when generalizing findings.

Action research has not been widely used in software engineering—in their survey of software engineering research, (Glass et al., 2002) found that 0% of the sampled articles used it as the dominant research method. Since the statistics are based on a sample, while not showing the total absence of action research, they clearly show that it is not common in our field. This is perhaps somewhat surprising, since action research can be used to alleviate the problems of practical relevance, validation, and naturally supports the iterative nature of design work. However, having read several articles in the field of software engineering that contain "case studies" validating various approaches, I tend to agree with Avison et al. (1999) that much research reported as "case studies" seem to have "action research characteristics", even if the actual term action research is not used.

The action research process has been described by Susman and Evered (1978) as consisting of five steps, as shown in Figure 3.2.

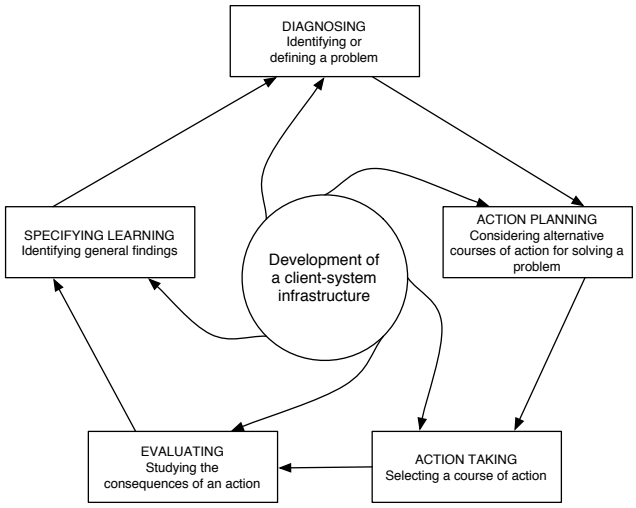


Figure 3.2: The Action Research Cyclical Process (Susman and Evered, 1978)

Central to the research process is the establishment of the client-system infrastructure, in which mutually agreed upon rules and forms for the collaboration are defined between the researcher and the subject of the research ("client"). The cyclical process starts with a joint diagnosis of the problem to be addressed, followed by action planning considering various ways of attacking the problem. Then, a course of action is selected and taken, after which the consequences are evaluated. Finally, general findings are identified, after which the cycle restarts from a diagnosis of the new situation. As discussed in the next section, this process closely described the way we worked, the main difference being the central role the constructs we were developing played in the research.

3.5 Research Process: "Constructive Action Research"

The research discussed in this thesis basically followed the process for constructive research discussed above, with the construction and validation taking place in action research cycles, of which there typically was several in each case study. Figure 3.5 illustrates the steps.

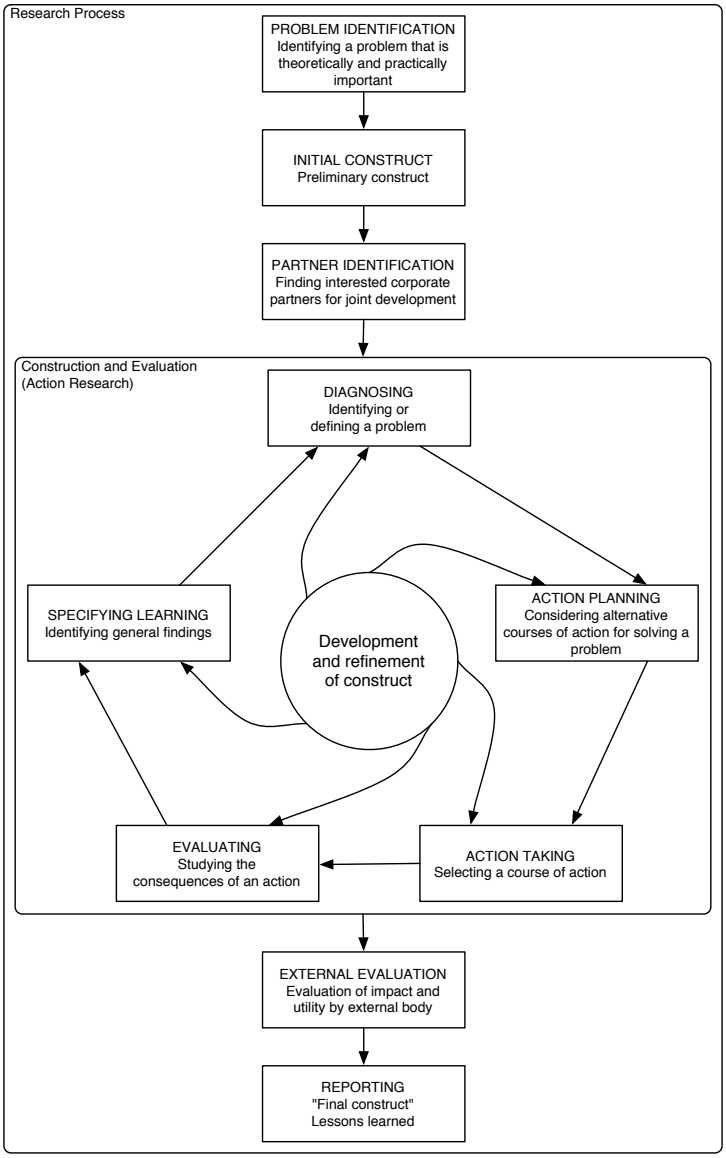


Figure 3.3: Research Process

3.5.1 Problem Identification

As previously explained, we started the research with a preconceived idea: to help companies improve their ability to control their software development operations by instituting visual, on-line feedback of status information in the form of control panels to different stakeholders, including both management and practitioners. We motivated this initial idea with the success of measurement-based approaches in manufacturing, the already quite widespread use of process management techniques in both product and software development, and our own earlier observations of the low practical industrial utilization of measurement in management of software development. An initial literature survey of the main approaches of measurement and process improvement in software engineering found no directly applicable framework or method or tool to use in our efforts. We thus think that the identified problem fulfills the criteria of both theoretical and practical relevance.

3.5.2 Initial Construct

Based on a literature study and our previous experience in working with Finnish software development companies, we developed a preliminary vision consisting of the idea proposed in the previous chapter: we envisioned software development status information being continuously available to all relevant stakeholders in role-specific and tailorable visual displays, so called *control panels* that were accessible from anywhere, automatically updated, and applied at all organizational levels. We recognized at this point that such control panels would only be the "tip of the iceberg", i.e., that their development would require substantial investments in measurement program development, implementation and tool support.

3.5.3 Partner Identification

In order to get corporate partners to work with, we approached twelve companies that we thought would be interested in working with us in defining and implementing visual control panels for their software development activities. We thus used what Patton (1990) would call *purposeful sampling*.

We approached the companies with a preliminary project plan and a presentation in which we outlined the main idea: to develop stakeholder-customized visual "control panels" for monitoring their development activities. The companies were to sign up for a three-year joint project, which was to be financed by the Finnish Funding Agency for Technology and Innovation (TEKES) and the participating companies. TEKES was to finance 75% of the project costs, and the companies 25%. In addition to providing project funding, we asked the companies to provide adequate resources, which we at this stage interpreted as meaning one full-time equivalent.

Of the contacted companies, all agreed that the problem we intended to solve was important, and that the proposed ideas were face valid. After negotiations, we had five companies that agreed to work with us. Since one of the companies

did not have any software development activities but wanted to apply the ideas to its project business[1], I have decided to exclude it from this work.

Table 3.2 presents the case companies and some basic characterizations. The business areas ranged from security systems development to telecom and business software, and the size of the companies ranged from 45 to 600 employees. Two of the companies, TeleCorp and SecuriCorp were subsidiaries of international groups. For these, the numbers reported in the table express the size of the Finnish subsidiary. The number in parentheses represent the whole group. Most of the development organizations (departments) were quite small, with the exception of TeleCorp's, which employed 130 persons.

Table 3.2: Case company overview

Case company	SoftCorp	TeleCorp	Electro-Corp	Securi-Corp
Business	Business software	Telecom	Ind. electronics	Security
Total # of employees	45	1000[a]	250	50
Size of development org.	7+3	130 (600[b])	60	11
# of simultaneous projects	1 / 10	3	20–30	5
Formal process / process age	N	Y/>10	Y/2	N
Typical project size	2 mm	30000h	?	?
Typical project duration (months)	6	9–12	6	24–36
Typical schedule adherance	+100%	+0–8%	+0–200%	+0–100%

[a] Finnish subsidiary
[b] Total in Finland

The number of simultaneous projects also varied greatly, with ElectroCorp having the most difficult situation, in which a small department tried to juggle an enormous amount of projects at the same time.

Two of the companies had a formal, institutionalized new product development process, whereas the others were in the process of defining and implementing formal processes. TeleCorp's process culture was well established, whereas the process at ElectroCorp was developed purely for the need to satisfy ISO-certification, and seemed to have little impact on the actual work performed.

The typical project size and duration varied a lot—both between and within companies, due both to differences between the companies, and to different project types within a single company. The projects with the longest duration were found at ElectroCorp, where the development of completely new products typically took up to three years, whereas the planned duration was between one and two years. The schedule adherance was poor in all companies except TeleCorp, in which the projects typically kept their schedules extremely well. Management at TeleCorp attributed this to the long-term focus on process improvement and

[1] The company developed physical prototypes using rapid prototyping technology.

measurement both in the Finnish subsidiary and at the group level. On process maturity, only TeleCorp had assessed its process, to level 2 of the Capability Maturity Model. The other partners had not made assessments, but were clearly at level 1.

In summary, the companies we worked with consisted of three relatively immature SME's that lacked, or were in the process of institutionalizing, formal processes and measurement systems, and one subsidiary of a large telecommunications group with a quite mature process culture. Thus, we had the opportunity to try out our ideas in a fairly wide set of situations. Next, we will look at the action research cycle used in the companies.

3.5.4 Action Research Cycle

The practical work in each company was tailored to the specific situation and needs of the firm in question, and followed a general action research cycle. From the point of view of developing control panels, we performed the related activities listed in Table 3.3. The activities were performed in one or several action research cycles, as described more fully in Chapter 6.

Table 3.3: Cases and activities performed

Case	Current state analysis	Meas. system def.	Meas. syst. impl.	Tool piloting	Post-mortem
SoftCorp	X	X	X	X	X
TeleCorp	X			X	X
ElectroCorp	X	X	X	X	X
SecuriCorp	X	X	X	X	X

In the case studies, we used a variety of data collection methods, as shown in Table 3.4. This is common in case study research, and is a basis of *triangulation*, which increases the reliability of the findings (Jick, 1979; Patton, 1990; Yin, 1994).

Table 3.4: Cases and Data collection methods

Case	Inter-views	Obser-vation	Work-shops	Docu-ments	Survey	Tool logs
SoftCorp	X	X	X	X		X
TeleCorp	X		X	X	X	
ElectroCorp	X	X	X	X		
SecuriCorp	X		X	X		

Next, I briefly describe how we approached the general action research activities during our work.

3.5.4.1 Diagnosis

We started each cycle by analysing the current state of the object selected for study. The analysis was done mainly by interviews and document analysis, but when appropriate, we also used observation. In the initial cycle for each of the cases, the researchers typically spent several days on-site gathering information and getting familiar with the company, its culture and processes. The current state analysis of each company was written down and checked by corporate representatives. Typically the reports also contained suggestions for improvement in addition to the analysis. These initial analyses led us to the development of the "general control system" framework as a way of viewing the "big picture", discussed further in Chapter 4.

Subsequent cycles used similar data collection methods, and typically focussed on some specific part of the above mentioned framework. As typical for action research, the exact data collection and dissemination practices varied according to the needs of the situation and the specifics of the client-researcher infrastructure.

3.5.4.2 Action Planning

In the action planning phase the researchers and company management jointly considered the results of the diagnosis phase, and evaluated what actions to take. Typically this consisted of deciding the particular parts of the organization or stakeholders for which control panels should be developed. In this stage, also the necessary actions were planned, i.e., whether to simply install the toolset or to additionally define measures, and data collection procedures.

3.5.4.3 Action Taking

The action taking step consisted of carrying out the actions decided upon in the previous phase. In the cases where we developed measures, we used mainly two approaches for measurement development: the GQM approach (Basili and Rombach, 1988) and the Balanced Scorecard (Kaplan and Norton, 1992). Where needed, we also developed the data collection procedures and forms, typically using MS Excel. In addition, we installed the toolset and built the necessary data transfer modules. The action taking phase was typically conducted jointly by the researchers and corporate stakeholders.

3.5.4.4 Evaluating

During the evaluation phase, the stakeholders involved in the action taking phase evaluated the actions taken, and their results and/or effects. The evaluations typically took place in workshop settings with all relevant stakeholders present.

3.5.4.5 Specifying Learning

Based upon the evaluation step, we as researchers identified lessons learned during the action research cycle. These lessons were fed into the next step and subsequently incorporated into our constructs.

3.5.4.6 Development and Refinement of Constructs

Finally, the lessons learned were considered from the point of view of the constructs that were developed. Where appropriate, we changed or added toolset features, or adjusted our approach for developing control panels.

3.5.5 External Evaluation

At the end of the action research case studies, we evaluated the practical impact of the research. Both we and an external consultant interviewed the companies we had worked with. The use of an external consultant can be viewed as one way of minimizing researcher bias.

3.5.6 Reporting

The final stage of the research consisted of write-up, the final version of which is this document.

3.5.7 Summary of Case Studies

The empirical studies are summarized in Table 3.5, which shows their respective objectives and main results. As shown in the table, the case studies had slightly different goals, and the activities performed were tailored to the particular situation of the company.

The case studies were not performed in synch: as shown in Figure 3.4, the SoftCorp and ElectroCorp cases started one year earlier than the SecuriCorp and TeleCorp cases. The timeline shows the various cycles in the case studies. These are discussed in more detail in Chapter 6.

3.6 Conclusions

This chapter presented the research approach used. First, I discussed at a general level approaches to research in software engineering. Following this, I presented the approach used in this work, "Constructive Action Research", which combines elements of design research and action research. Finally, I described the research performed in this thesis according to this process, as well as presented a general overview of the steps taken and the contributions of the case studies.

Table 3.5: Summary of Empirical Studies

Study	Objectives	Results/Contributions
Case 1: SoftCorp	• Explore the control panel idea in an SME setting • Build and test control panels at the project, and strategic levels	• Analysis framework • Approach feasible also in small companies/projects • Data and experiences on toolset deployment • Data and experiences on small-scale control panel usage
Case 2: ElectroCorp	• Analyze the product development process • Develop and install visual control panels for product development	• Explosive growth of number of goals and measures • Experiences with corporate resistance, politics and fear of measurement • Understood need for visualization libraries
Case 3: SecuriCorp	• Build and install control panels and measurement system for a small software development organization within a large company	• Experience with project classification and portfolio management • Toolset piloting experiences • Understood need for visualization libraries
Case 4: TeleCorp	• Analysis of existing measurement system • Piloting of toolset using existing measurement infrastructure	• Goal-measure mappings • Toolset features: remote queries and proactive caching • Toolset piloting experiences

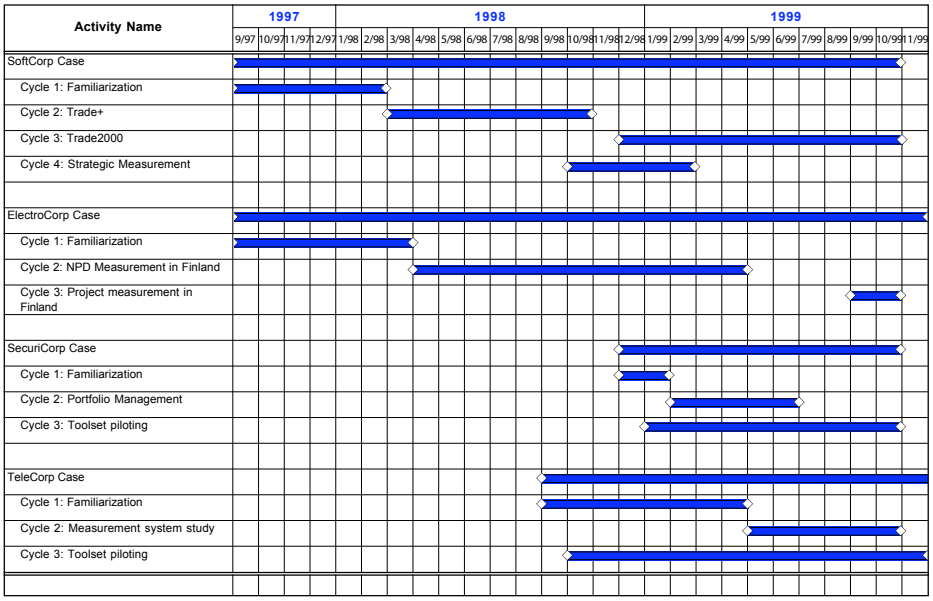


Figure 3.4: Case Study Timeline

Chapter 4

Software Development Control Panel Concepts and Guidelines

> You cannot control what you don't measure.
>
> Tom DeMarco

4.1 Introduction

When developing control panels, we basically need to tackle three issues: what panels to develop, for whom to develop them, and how to integrate the development effort with a corporate measurement program. In this chapter, I first restate the requirements for framework development and define the concepts used. Then, I discuss ways of partitioning the control panel development effort. Finally, I discuss how to integrate the control panel development effort with the corporate measurement program.

The discussion in this chapter presents the "final" version of the constructs, which as such has yet to be validated. As described earlier, the concepts were developed iteratively using several concurrent action research cycles in the case companies. This chapter thus reflects our understanding, based on the empirical experiences, on how to structure and develop control panels. Earlier versions of the constructs have been published in (Lassenius et al., 1999) and (Lassenius and Rautiainen, 1999).

4.2 Requirements

As mentioned in the introductory chapter, we developed a list of requirements for the guiding conceptual framework. These are shown in Table 4.1.

The first requirement states that the central concepts for developing control panels should be developed. The idea was to develop a minimal sets of concepts needed, and as far as possible use existing terminology from the literature.

The second requirement calls for guidelines for developing control panels.

Table 4.1: Requirements for framework and guideline development

No	Requirement
FR-1	*Concepts:* The framework should define the central concepts needed for the development of control panels.
FR-2	*Guidance:* Guidelines for control panel development should be developed.
FR-3	*Method independence:* Control panel development should be independent of measurement paradigm or methodology.
FR-4	*Organizational independence:* It should be possible to develop control panels for different organizations and environments.
FR-5	*Technological independence:* Control panel development should be independent of the particular technological tools deployed.
FR-6	*Usability:* The framework should be easy to learn and use.
FR-7	*Feasibility:* Control panel development should be feasible in practice.

Requirements FR-3, FR-4 and FR-5 refer to the generality of the constructs. We envisioned developing control panels in various organizations with different levels of existing measurement systems and tool support. Thus, we aimed for methodological, organizational and infrastructural independence.

The final two requirements are related to implementation concerns: we wanted to make our ideas easy to implement by keeping them as simple as possible, and to make them as cheap to implement as possible.

In practice, we addressed the concerns for developing control panels using three constructs: a conceptual framework linking the concepts of *visualizations* and *control panels* to concepts typically found in corporate measurement programs, a discussion about what panels to build and for whom, and a set of activities that needs to be performed when developing control panels. The following sections describe these constructs.

4.3 Concepts for Software Development Control Panels

We developed a simple conceptul framework tying the control panel idea to the concept of a measure. The concepts and their relationships are depicted in Figure 4.1.

As shown in the figure, a *control panel* consists of one or more *visualizations*. A *visualization* is a graphical representation of one or more *measures* and its possible *target values*. Each measure can be depicted in several visualizations, and each visualization can be used in several panels. Visualizations can also contain *links* to panels, enabling navigation between panels. Links can be parameterized to enable context-dependent navigation. This makes it possible, e.g., to link a certain activity in a Gantt-chart to a panel showing more detailed info on that particular activity. Non-context sensitive navigation can be implemented using *buttons* that simply provide a static link to a predefined panel.

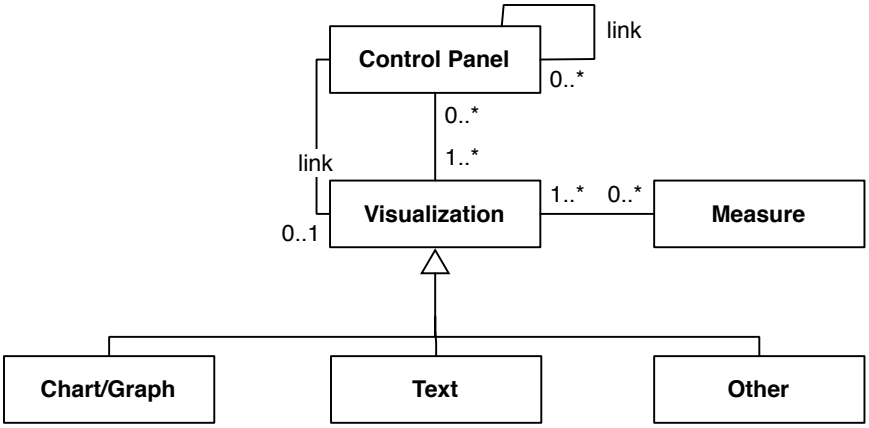


Figure 4.1: Control Panel Concepts

Measurement is always related to something that is being measured, a *measurement object.* A typical classification given in the software engineering literature divides measurement objects into processes, products, and resources, i.e., we can measure the processes and activities related to software development, the actual products (artifacts) being developed, and we can measure the resources (people, machines etc.) doing the development. In the case of resource measurement, the object of measurement typically at the same time represents a role and a stakeholder to measurement.

From the point of view of developing control panels, taking into account the requirement for measurement paradigm independence, our conceptual framework does not go deeper into the contents of the measurement program itself, but interfaces at the measure concept. We thus expect that a measure that can be visualized has defined and instantiated data collection, validation, and analysis procedures.

4.4 Defining Control Panels

4.4.1 Deciding What Panels to Build

Having described the simple conceptual framework for control panel development, we are now ready to discuss the difficult question of what panels to develop and for whom. This questions is intimately tied to the development of the corporate measurement program, since the measures available dictate what can be included in the panels.

In most practical cases though, the process is likely to be iterative, with control panel definition and development also influencing the measurement program.

When deciding upon what panels to develop, one must strike a balance between on the one hand the need for customization to the needs of various

stakeholders and characteristics of, e.g., a particular development effort, and on the other hand the need for cost effectiveness and ability to benchmark between measurement objects.

I think that the development of completely universal control panels, like the one by the software project manger's network, is unlikely to be successful[1], since the details of process, context, resources and products tend to be specific to a particular organization. This, I think, means that the specific visualizations and their combinations into control panels need to be customized to the organization for which they are developed. In addition, there is likely a need to customize panels even within one single organization, e.g., depending on the type of development being done.

The answer to the question of what panels to develop is intimately intertwined with the question of how many, and which, measurement programs to develop. There is a wide variety of possibilities for structuring a measurement program. The conceptual separation between visualizations, control panels and measures gives us some additional leeway. For example, one could imagine a situation in which the same measure is taken, but visualized differently to different stakeholders, e.g., showing aggregate visualizations to top management and details to software practitioners. In a typical situation in which measurement development and data collection might require some manual labor, but in which visualizations and control panels are automatically generated by tools, this means that we can build customized visualizations and control panels that can be used without excessive additional cost.

The following list provides different ways of partitioning measurement into control panels:

- **Roles**. Separate control panels are developed for each role, such as project manager, process developer, software developer, tester, process owner, etc.
- **Stakeholders**. Each person with a vested interest in some aspect of software development gets his or her own panels. In many organizations, this effort can start with developing panels based upon stakeholder roles, that later can be customized to individual stakeholder interests
- **Organizational level**. Different panels are developed for different organizational levels: etc. strategic, operational, process, project, team, individual.
- **Process Phases or Activities**. Panels are developed that highlight particular steps or activities in a process, e.g., design or testing.
- **Project types**. Different panels are developed for different project types, such as maintenance, new product development, or bugfix (service-pack) projects.

[1]This is in line with both the more general viewpoint taken by the Goal-based measurement school, as discussed in Chapter 2, and the experiences with the SPMN control panel (Anonymous, 2000b), the manual of which states that the most significant change to the tool in its subsequent versions has been the addition of possibilities for customization of the panels.

- **Purpose**. Panels can be developed for different purposes: for monitoring, for learning, prediction, etc.

I do not claim the above list to be exhaustive, and its elements are certainly not orthogonal. I do, however think that it covers the most typical scenarios. In practice, any particular effort is likely to use a combination of the listed possibilities.

In addition to the above list, we developed a framework of a *generic control system*[2] for project organizations, which is depicted in Figure 4.2[3]. We found it easy to communicate this framework to our corporate partners, and found it fruitful when discussing what panels to build, and for what purpose.

Figure 4.2 shows one way of viewing a project-oriented software development organization. In the software engineering literature, much focus has been given to the *software process*, and *software project management*. Much less—if any—attention has been paid to the higher-level views of strategy, portfolio management and competence management. While, e.g., competence management is outside the scope of this work, I have included it in the framework, since we have found the discussions on competence management and measurement to be fruitful in the organizations we have worked with, and because software development is an activity that is strongly linked (or should be) to competence management. The purpose of this presentation is not to go into any detail on any of the topics mentioned, but to serve as an illustration of one way to partition the world when planning the development of control panels.

As indicated by the references in the subsequent paragraphs, that describe the framework, the body of knowlegde on which the concepts rely is not from the field of software engineering, but from new product development management, a field which is typically infrequently referred to in software engineering literature, but with which there clearly is room for cross-learning and cross-fertilization of ideas (Nambisan and Wilemon, 2000).

4.4.1.1 Software/Product development strategy

By strategy we refer to the product development strategy of the company, which should be derived from the overall corporate strategy. An important issue is to understand that there are different types of product development projects that need to be staffed and managed in different ways. For example, developing a new piece of software is different from making incremental maintenance releases to an already existing product. If product maintenance is considered part of development, it is also managed differently. The development strategy can be summarized as one or multiple roadmaps (product, service, marketing, etc.), where for instance the product roadmap should show the different types of projects and a rough resource allocation. This is used as an input to project portfolio management and competence management. (e.g. Cooper (1993); Cooper et al. (1998)).

[2] I use the term *control system* to mean *the formal, information-based routines and procedures managers use to maintain or alter patterns in organizational activities* (Simons, 1995)

[3] A previous version of the generic control system shown here has been published in (Rautiainen et al., 2002). In this article, the framework is mentioned as having been developed earlier. I developed the initial version of the framework, as well as made the changes resulting in the one shown here.

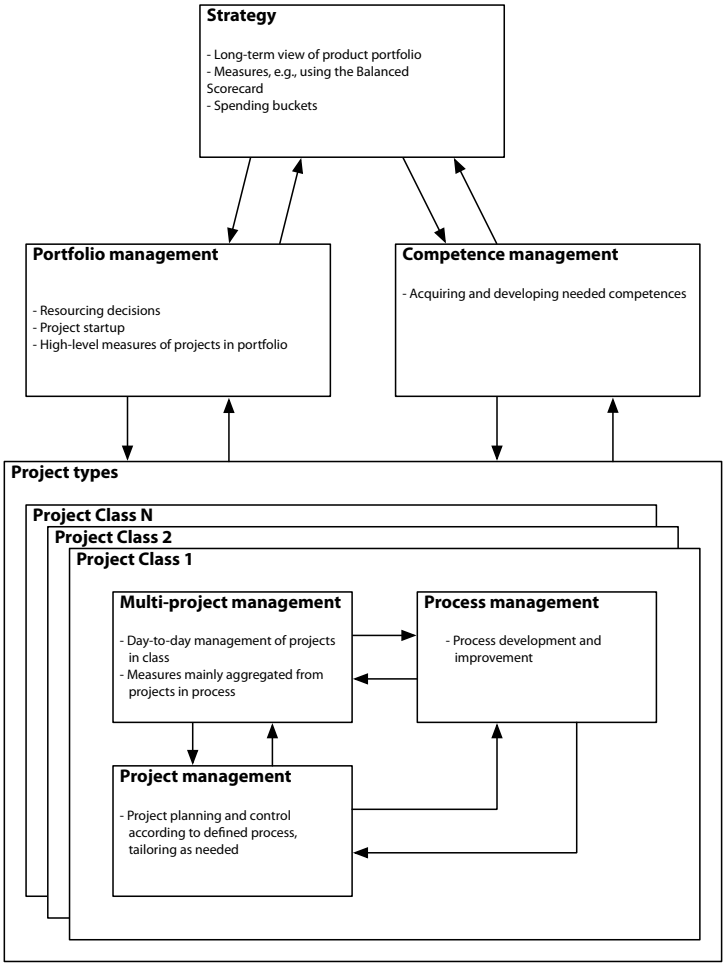


Figure 4.2: A Logical View of a Project-Oriented Software Development Organization

For examples of project classifications, see e.g. Shenhar (1998) and Wheelwright and Clark (1992).

4.4.1.2 Portfolio management

Portfolio management refers to the management of the whole set of development projects. The input to portfolio management is the product roadmap, especially the project type classification and the rough resource allocation. To be successful in portfolio management, one must also know the existing resources and competences in the organization. Another input is, of course, the feedback from ongoing projects. The purpose of portfolio management is to specify in more detail the projects needed to fulfil the strategic goals of the organization, thus linking projects to strategy and operationalizing the product roadmap. An important task

is to prioritize projects and select the order and mix of projects to be executed. The output of portfolio management is an aggregate project plan or a project roadmap. The plan has to be updated at regular intervals to reflect the current situation. For example, many projects can be interrelated and if one project is lagging in its schedule, other projects can be influenced. This may lead to replanning and reprioritizing the order and mix of projects. For more reading, see for example (Englund and Graham, 1999) or (Cooper et al., 1998).

4.4.1.3 Process management

Process management is about managing the development process. It relies on a process model that works as a map for the development projects providing the stages, milestones, roles, etc. It provides a common vocabulary and the "rules of the game", i.e., how things are supposed to be done in the organization. The process model should also be a tool, providing, e.g., templates and checklists for the projects. It is important to realize that one process model cannot accommodate the needs of all different project types. Therefore, some thought has to be put into choosing appropriate models. Some examples of issues affecting the choice of process model are the speed of change in technology or the markets, the size and length of the projects, the size and complexity of the product being developed, and the initial uncertainty of the project, i.e. how well we know the requirements up front. One part of process management is collecting data and feedback from projects for process improvement purposes.

4.4.1.4 Multi-project management

The purpose of multi-project management is to balance and allocate resources between projects at a regular and short-term basis. Having people work on multiple projects and moving them around between projects is not easy, though, and can cause more harm than gain (Brooks, 1995; Smith and Reinertsen, 1998). Multi-project management is, naturally, closely linked to project portfolio management, and could even be considered part thereof.

4.4.1.5 Project management

Project management is about executing the individual projects in a systematic way, using the guidelines provided from the process models. Software project management is a subject well covered in the software engineering literature, and is taught in most software engineering programs. Software project measurement experiences have also been reported in the literature, e.g. (Grady and Caswell, 1987; Grady, 1992; Daskalantonakis, 1992).

4.4.2 Applying the control system framework: A Fictive Example

As an illustration of applying the control panel framework, consider Figure 4.3. The figure shows the control system and Figure 4.4 related panels for a fictive com-

pany, SoftGuys Ltd. Product strategy is visualized using a roadmap for product introductions. Portfolio management is visualized using a Gantt-chart depicting the aggregate project plan.

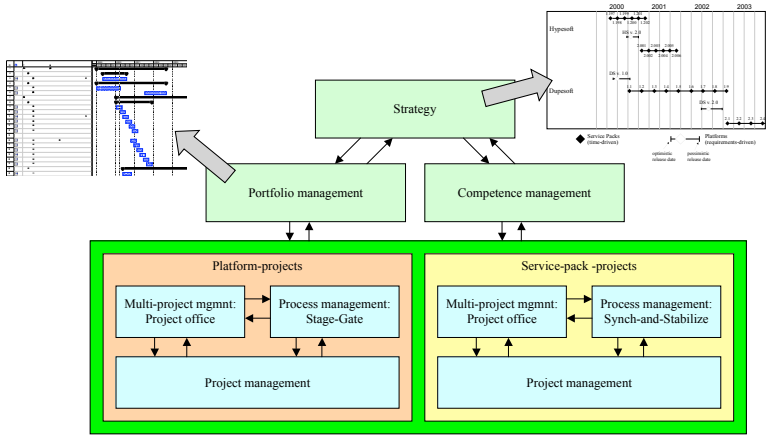


Figure 4.3: The SoftGuys Ltd. Control System

The company has two distinct project types: one for developing new product platforms, and one for delivering incremental changes ("service-packs") to existing platforms. The platform project process has a fixed scope, and follows a typical Stage-Gate (Cooper, 1993) model, which is similar in many ways to the waterfall model for software engineering (Royce, 1970). The service-pack process follows the synch-and-stabilize model (Cusumano and Selby, 1995). Multi-project management for both processes is handled by a joint project office.

Figure 4.4 shows one possible set of control panels that could be developed for the company. The panels have been developed according to the control system model, and shows one strategic panel, and different panels for multi-project management, and project management respectively, depending on which process one wants to look at.

The arrows in the figure indicate navigation. In the strategic panel, one can select one of the processes for further study. This illustrates the use of static navigation using buttons. The multi-project panels allow for parameterized navigation: they both contain a Gantt-chart. By clicking a bar—which represents a project—one can open a detailed view of that particular project.

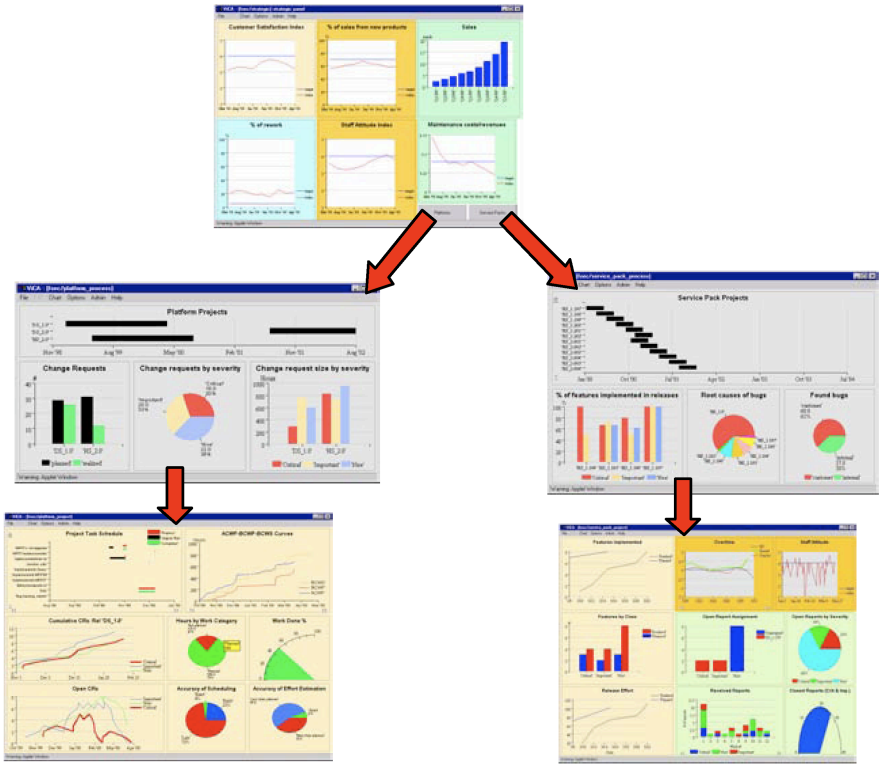


Figure 4.4: The Control Panel Hierarchy at SoftGuys Ltd.

4.4.3 Guidelines for Control Panel Construction

Having made a high-level decision regarding what panels to develop, we can next turn our attention to the development of a single panel. While the measurement literature contains methods and advice for developing and implementing measurement programs, little advice is available on the grouping of measures into panels, I here present three guidelines that might provide useful.

1. **Aim for a balanced panel, considering different perspectives.** In the same way as overemphasizing a single measure can be dangerous, so can considering only a single perspective. When building panels, it can be worthwhile to consider including measures of different perspectives. The field of software measurement contains several classifications of measures, e.g., into product, resource and process measures (Fenton and Pfleeger, 1997). The SPMN project control panel's (Brown, 1996) five perspectives is probably often more than sufficient. Likewise, the Balanced Scorecard (Kaplan and Norton, 1992) contains perspectives that can be useful.

2. **Use the same aggregation level.** Many control panels will contain measures of a single object of interest, e.g. a project or a single process. Others can contain, e.g., views of several different processes in one panel. Since the

aim is to build a panel hierarchy, it is mostly a good idea to try to keep the measures in a single panel at the same aggregation level. For example, details of particular project activities might be better shown in a "task control panel" than on a project level panel.

3. **Consider single role requirements.** A good way of keeping the design of a single panel focused is to build it to meet the needs of a single role or stakeholder.

4.5 Control Panel Implementation Activities

The final question that we need to address, is how to go about defining the measures and control panels. Since control panel development is tightly related to measurement system development, the detailed process for developing them and the related measurement program is highly dependent upon the chosen measurement system approach. Therefore it is hard to provide a clear-cut process for the development of control panels. Instead, I here try to identify and at a high level describe the activities that must be performed in order to build a working set of control panels. The way activities get instantiated and performed is likely to vary for each organization.

4.5.1 Define Panels and Panel Hierarchies to Build

The first activity that must be performed is to decide what control panels to build. As discussed above, there are several ways to partition the effort.

4.5.2 Define Panel Visualizations

After selecting the panels, each panel must be defined. This is best done by having a measurement expert work together with the stakeholders for whom the panels will be built. While there is a lack of elicitation methods (Münch and Heidrich, 2004), a discussion or brainstorming centered around the information needs of various roles and stakeholders can be effective. It is useful to have the future control panel users draw candidate visualizations. Following this, the visualizations can be grouped into panels. This can be done, e.g, using post-it notes on a whiteboard or wall. Navigation can also be illustrated using, e.g., arrows between the panels.

When the panels have been designed, their information (data) needs should be defined. The indicator template of Goethert and Hayes (2001); Goethert and Siviy (2004) can be useful to this end.

4.5.3 Consolidate and Integrate with Existing Measurement Program

When the control panels have been designed and their information needs determined, the information needs should be consolidated. The consolidated data

requirements and their possible mismatch with the existing measurement program must be reconciled. For example, if some data needed by the planned panels is not available, new measures must be defined and implemented, or the panels changed.

4.5.4 Build Infrastructure

The control panels demand a working support infrastructure. In addition to the measurement program's tools, it is probably necessary to install specific tools for automatic generation and visualization of control panels, e.g., the Lucos toolset described in the next chapter. This entails selecting and installing the needed tools, and instituting the necessary technology to make data from the measurement program's databases visible to the control panel tools. This can require installation, or implementation of tailored data transfer tools, like the EDAMs discussed in conjunction with our toolset.

4.5.5 Implement Control Panels

When the infrastructure is in place, the control panels need to be implemented and tested. Our experiences, discussed in Chapter 6 show that particular care needs to be taken with making sure that individual visualizations get and show correct data. It is also suggested that the control panels be piloted before taken into large-scale use.

4.6 Conclusions

This chapter presented a conceptual framework for developing control panels, presented ways of partitioning a measurement program into control panels, including a framework showing a high-level view of a typical software development organization, and finally identified the activities that need to be performed when developing control panels for software development.

Chapter 5

A Measurement Toolset

Don't Automate—Obliterate!

Michael Hammer

5.1 Introduction

In order to get experiences with the implementation and use of control panels, we implemented a prototype toolset for use in our client companies. This chapter presents the motivation, architecture and functionality of this toolset. The chapter is partly based upon previously published articles that I have coauthored (Vanhanen et al., 1999, 2000).

5.2 Motivation

In the software engineering literature, the need to support organizational measurement with suitable tools is often stressed (e.g. Grady and Caswell (1987); Basili and Rombach (1988); Tian et al. (1997); Kempkens et al. (2000)). Proposed and realized benefits of supporting measurement with tools include lower cost and increased reliability of data collection, faster feedback cycles, easier reuse of measurement related procedures and metrics; and increased acceptance of measurement. Since our approach to visual control relied on fast feedback with tailored visualizations to the needs of different stakeholders, we felt that tool support was necessary in order for the approach to be feasible.

5.2.1 Tool Requirements

Based upon our initial vision, and discussions with our case companies, we derived a set of initial requirements for the toolset. The first three requirements followed directly from the vision as explained earlier:

Requirement 1 *The system should support the definition and display of customizable control panels for different stakeholders.*

Requirement 2 *The system should support navigation between different panels.*

Requirement 3 *The system should support authentication and access control*

The need for classifying data into *private* and *public* is stressed by Grady (1992) and viewed as a requirement for successful measurement program implementation. Therefore, access control is necessary. Software process approaches, such as the Personal Software Process (Humphrey, 1995, 2000a) and the Team Software Process (Humphrey, 2000b) also suggest this.

Requirement 4 *The system should be compatible with existing corporate measurement systems and data sources.*

Although not incorporated into systematic data collection and validation procedures in measurement programs, corporate data is today largely stored in electronic form in existing databases and information systems, e.g., spreadsheets and configuration management systems (Crnkovic and Willfor, 1998; Sprague, 1991; Lavazza, 2000). Much of this data can often be—either directly, or by applying suitable transformations—be used when developing software measurement programs, and—in our case—control panels. From the point of view of implementing measurement, it is important to be able to interface to these systems since data collection and storage tend to be the most expensive measurement activities (Fuggetta et al., 1998) and requiring the organization to enter the same data into several systems is unfeasible.

Requirement 5 *The system should support different approaches to the definition and application of measurement.*

Since the system is intended to support measurement at various corporate levels, and in a variety of settings, tying it strictly to any specific measurement paradigm is limiting. For example, whereas GQM (Basili, 1984) is widely touted as the best paradigm for developing software engineering measures, other approaches, such as the Balanced Scorecard (Kaplan and Norton, 1992) are popular at the strategic level. In order to support corporate wide measurement it is therefore necessary to support different paradigms, as forcing the whole organization to use a single measurement paradigm in our view is unfeasible in practice.

We also envisioned a system which would work independently of physical location, a requirement important in particular to TeleCorp and ElectroCorp, which both had several development offices.

Requirement 6 *The system should support people working in physically separate locations.*

Since we envisioned that the system be used by lots of people at various corporate levels and with various degrees of computer literacy, we required that

Requirement 7 *Basic system usage should be easy and not require special skills or training*

Our partners also worked on different computing platforms, leading to the need for a cross-platform solution.

Requirement 8 *The system should work in both Windows and Unix environments, and must support both the Netscape and Internet Explorer browsers.*

Since some of our clients were SME's, we had to make sure that the toolset would not be too costly.

Requirement 9 *The system must be inexpensive*

This final requirement excluded, e.g., the use of expensive commercial databases. Finally, it is worth noting that the purpose of the development of the toolset was primarily to do a proof-of-concept prototype useful as a research tool for deepening our understanding on designing and implementing control panels in industry. To this end, we added a requirement that is mainly of interest from a research point of view:

Requirement 10 *The system should support logging of user actions*

We envisioned that we could, by analysing the logs, get more information on both normal and administrative usage. Regarding basic usage, we were mainly interested in knowing, e.g. what panels were used, by whom, and how often. Administrative usage was interesting in the sense that we could get information on the amount of time administrators spent on building single visualizations and control panels.

Based upon the requirements, we defined an architecture, which is discussed in the next section. Subsequent sections describe the architectural components in more detail.

5.3 Architecture

While there are several possible architectures that can help meet the requirements decribed in the previous section, we opted for a three-tier architecture using Web-technologies, as a simple and cost-efficient way of meeting the requirements. We decided to circumvent the difficult problem of making a generic design for a measurement database (see e.g. Foltin and Dumke (1998)) by deciding to use a replication strategy; we simply provide an application program interface (API) using which the organization can develop tool that replicate data from existing sources into a central database used by our visualization system. Thus, issues of data replication strategies, data consolidation etc. are scoped out and left to the organization.

The tool architecture that we designed, shown in Figure 5.1, consists of external data transferring and replication modules (EDAMs), the metrics server (MESS) and applications utilizing the collected data, e.g., the Visualization Client Applet (ViCA). MESS stores collected data, performs user authentication and access control, and provides services for manipulating metadata. All client programs, i.e.,

EDAMs and client applications, communicate with MESS over an Intranet or the Internet using an HTTP based protocol (Berners-Lee et al., 1996). As the communications protocol is based on HTTP, the system is essentially location independent, and easily extensible, e.g. to provide secure transport using built-in or third-party solutions, such as SSL or SSH.

The next sections describe the components of the toolset in greater detail.

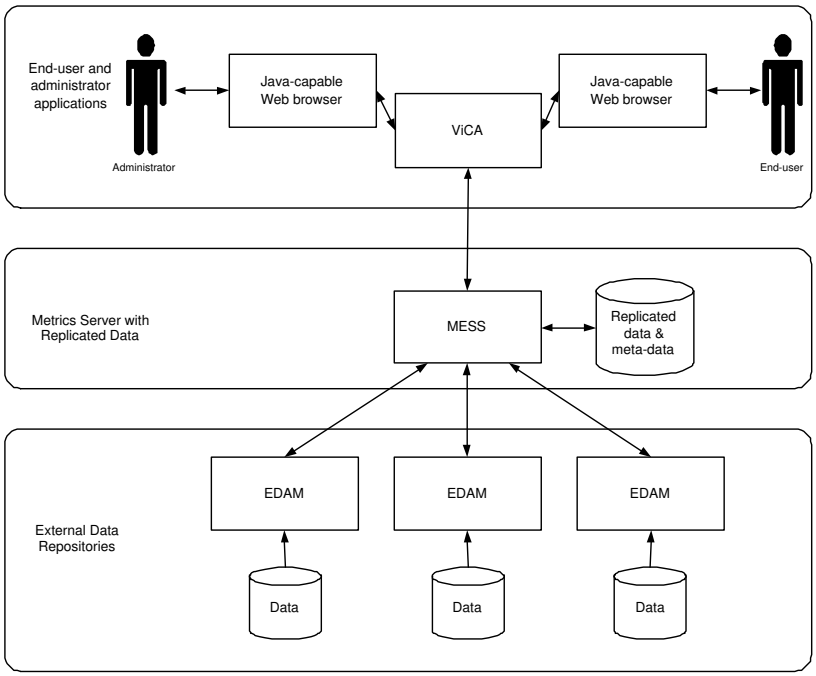


Figure 5.1: The Lucos Toolset Architecture

5.4 ViCA - The Visualization Client Applet

From a user or administrator point of view, the most visible component is the Visualization Client Applet (ViCA), which is used both for defining and creating visualizations and control panels. We implemented ViCA in Java in order to be able to support various operating system platforms. ViCA was designed to be as browser independent as possible, which at the time meant supporting both the Netscape and the Internet Explorer browsers.

There are basically two user categories using ViCA: ordinary users, and administrators. Ordinary users use ViCA to display measurement data, to create new control panels using existing visualizations, and to tailor the own control panels. Administrators use ViCA to create new visualizations.

5.4.1 Basic Usage

Viewing predefined control panels is similar to browsing the Web. ViCA is accessed by clicking a link on a Web-page, or simply by navigating to a specific page, if the application is run in Applet-mode.

ViCA's displays consists of *control panels* that contain *visualizations*, such as graphs and charts, as well as navigation buttons and images. The charts are graphical displays of data fetched from the server. ViCA supports fourteen chart types, including line, bar, area, pie and Gantt charts. The contents of a chart can be scrolled, which allows users to display large data sets in a single chart without cluttering it. Visualizations can also be *navigable*, which means that clicking on them (or a dataset in a chart) opens a new control panel, typically showing more detailed information about the selected item. Figure 5.2 illustrates the concept of navigation. In the figure, the source panel "All Projects" contains a Gantt-chart view of a set of projects. By clicking on one of the bars in the Gantt-chart, the user has opened a more detailed view of the selected project.

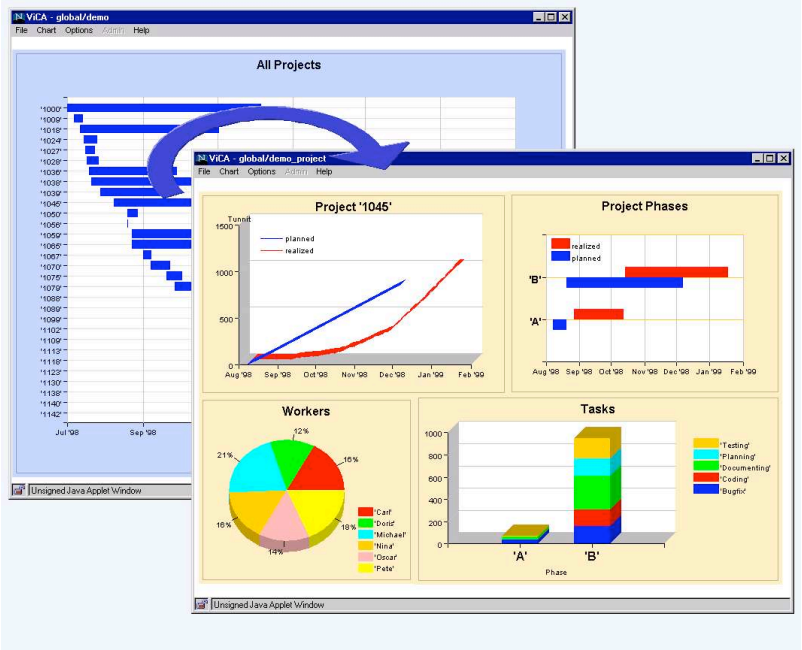


Figure 5.2: Navigation in ViCA

In addition to viewing charts, ordinary users can tailor attributes of existing panels according to their own needs. For example, changing the coloring, location in the panel, or the sizes of the charts is easy. Constructing new panels from predefined charts is simple, and can be done by normal users. Users are also allowed to make new charts, though this is somewhat involved since it requires SQL knowledge, and in our experience more suitable for administrators. In the small-scale practical use of our toolset, no user in practice developed any own

visualizations. On the other hand, they made small changes to both panel and chart layouts in a matter of seconds, as discussed in Section 6.2.5.4.

5.4.2 Administrative Usage

Administrators use ViCA to create new visualizations. This requires a working knowledge of ViCA, the structure of the MESS database, and SQL. Data for the charts is fetched from MESS using standard SQL queries. The mapping of a query result to a chart is flexible and allows the visualizion of several data sets, e.g., information on several projects, in a single chart.

Navigation is facilitated by *parameterized queries*, which allows the insertion of variables in SQL queries. The query is parsed during navigation. This alleviates the need to create several similar panels for the same purpose. For example, a general parameterized project panel may be constructed that shows information on a project that was selected for navigation in a project portfolio panel, as shown in Figure 5.2.

5.4.3 Visualization Libraries

Our initial experiences with industrial application of the toolset indicated that chart definition using SQL was difficult and required special skills not necessarily found in the companies. We therefore decided to implement collections of predefined visualizations, *visualization libraries* that contained common visualizations for use in various organizational settings.

The visualization libraries contain common domain specific visualizations and data requirement definitions. The libraries are added to ViCA as external components depending on the needs of the users. A domain specific library allows a user organization to know the minimum set of data required for practical visualizations and get the basic visualizations in use immediately. All chart definitions in a library are built on an SQL view interface.

The installation of a library requires this interface to be mapped to the corporate database. Thereafter a simple "Chart Wizard" can be used to create new charts based on the templates in a library. Charts are typically grouped into proper panels providing information on, e.g., a process or a project, or some aspect of them such as schedule monitoring.

5.5 The Metrics Server

The Metrics Server (MESS) is a central component in the framework, providing client programs with a variety of services. The services can be divided into basic services, logging, and remote queries and proactive caching.

5.5.1 Basic Services

The basic services provided by MESS include persistent storage facilities, user authorization, and data querying and manipulation. In addition it can operate as

a simple www-server. MESS stores and manipulates both the data collected from the data sources and metadata such as users, groups, schemas, views, charts and panels. User and group information is required for authentication and for limiting access to any object in the database on a group level. Schemas describe the field names and data types of database tables in the server database. The collected data is stored in these tables.Views are user generated virtual table descriptions typically used for simplifying complex queries. Charts and panels are objects generated and used by the Visualization Client Applet. Charts objects contain the SQL queries, data mappings and appearance of visualizations. Panels group related charts together.

In addition to entity specific attributes, each framework data entity contains also common metadata: name, description, and read, write and admin group name. MESS allows client programs to create, read, update, and delete entities and manipulate their metadata. Data can be dumped to user created tables and SQL queries executed to read the data. MESS can store and return files, thus serving as a simple www-server.

5.5.2 Logging

MESS also provides logging facilities that can log both client requests as well as internal events.

User actions can be logged at a detailed level—all requests from a client to the server can be put into the log file. This data is useful for several purposes. First, it can be used to get usage profiles for various users, making it possible to better understand how the system is used in real life: which visualizations are most popular, which paths are followed, how long various tasks take to complete etc. The use of logfiles for this purpose is illustrated in Chapter 6 in which we discuss our empirical experiences in using the toolset. ViCA also contains a panel visualizing the use of the tool itself. Second, the information in the log file can be used as an aid in enhancing the usability of ViCA, since also events like "Cancel" are logged. User events might even be reverse-engineered for the purpose of better understanding typical usage scenarios.

Logging of internal events are used by the proactive query caching mechanism described in the next section, as well as for debugging and performance improvement purposes.

5.5.3 Remote Queries and Proactive Caching

Sometimes source databases may be so huge that data replication to MESS is unfeasible. If this is the case, replication can be avoided by connecting MESS to a database management system that is capable of accessing remote, heterogeneous data servers, such as the Enterprise Data Studio from Sybase Inc. (Anonymous, 1998a). From the viewpoint of MESS, the difference is noticed only in slower response times to SQL queries.

Remote queries can, however, be prohibitively slow, as discussed in the TeleCorp case in Chapter 6. To help alleviate this problem, we implemented a tech-

nique we called *proactive caching* (Risku and Rinta-Aho, 1999) of commonly used SQL queries to minimize waiting times. Usage logs are used to find out usage patterns such as which are the most common SQL queries, how long they typically take to execute, and when they are executed. Based on this information, MESS tries to maintain valid results for common queries in its cache by executing the queries using processor idle time. Because MESS relays SQL queries from clients to the server database, it can return the result from its cache, if available. A problem with this technique is that without implementing push-technology (i.e. using triggers) in the source database, the cached query result might be out of date. In practice, since measurement data updates take place infrequently (i.e. not for example several times per second), this is unlikely to pose problems. It is always possible to manually update the queries to ensure that the latest data is used.

5.6 External Data Modules

Measurement data is transferred into the system by *external data modules*, or EDAMs. These are typically company-specific programs that connect to an existing corporate data source (such as a database, log file, spreadsheet, configuration management system, etc.), perform some optional data transformation, e.g., filtering, cleaning and reformatting, and then copy the relevant data to the server. Since an EDAM must connect to a company data source, the choice of architecture and programming tool or language is limited by the methods of access to the data source. Different EDAMs can be developed on different architectures with different programming tools as needed.

Typically, an EDAM performs the following steps when executed: First it establishes a connection to MESS and authenticates itself to gain access to target tables on the server database. It checks if the required database tables are present in MESS, and if needed, creates the missing tables and gives them reasonable access rights. After initialization, the EDAM connects to one or several data sources and fetches (or parses) the data and performs optional filtering, transformation or aggregation of the data. If possible, the EDAM only reads the new or changed data thus minimizing the amount of data to transfer to MESS. Finally, the source data that has been read and processed is dumped to the corresponding tables in MESS. It is worth noting that since measurement programs and thus also the visualizations and control panels used typically are dynamic in nature, i.e. new measures are introduced and old ones dropped, EDAMS should if possible be implemented with this in mind. Otherwise they might generate significant inertia to necessary measurement program changes.

Usually administrators of the system should write EDAMs to solve the organization's specific data transformation needs. However, we have implemented a programming library and a couple of EDAMs to help minimize the initial effort in deploying the system.

We implemented a C-language utility library that provides high-level functions to communicate with MESS. These include, e.g. initializing and closing connections, creating and deleting tables, and dumping data to a table. We also

implemented "CSVTool", a sample program for using the library that is also a useful generic EDAM that reads files containing data in CSV (comma separated value) format and dumps it to a specified table in MESS. CSV is a format that many applications, including MS Excel and MS Project, can generate.

Because most of our pilot users used Microsoft Project and Microsoft Excel for project management purposes we developed plug-ins for them as well. The Excel Metrics Information Exchange (EMIX) plug-in, shown in Figure 5.3 and the respective MS Project counterpart Project Metrics Information Exchange (PMIX) automate data transfer from the applications to MESS. They can be configured for each document separately to dump selected data in the document to specific tables in MESS. Thereafter they can make the dump automatically whenever a user modifies and saves the document. For example, when a new spreadsheet document template is created, the author marks in Excel all areas containing data to be transferred and specifies for each data area the table where the data is to be dumped. After this any user may make a copy of the document template for personal use and EMIX will transfer the data automatically when saving the document.

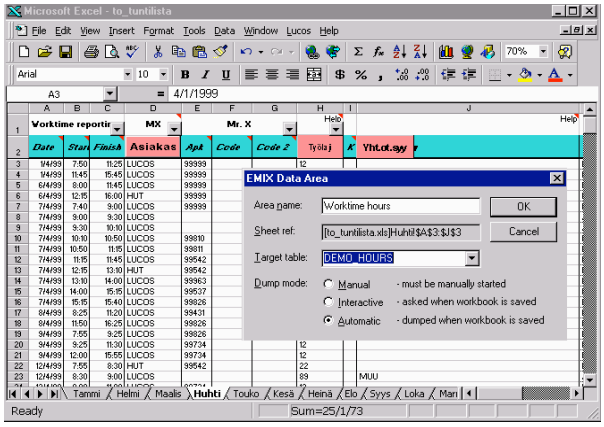


Figure 5.3: The Excel Data Module—EMIX

The mapping between the columns of a data area and the target table fields is automatic if the header line of each area contains the respective table field names. Otherwise, the user can manually configure the mapping for each data area. A data area may be appended with constant value columns. A constant is defined either as a reference to a worksheet cell or as a literal value. The constants can, e.g., be used to qualify time reporting hours by a person without the need to have the username duplicated on each row in the worksheet.

5.7 Conclusions

This chapter presented the Lucos toolset: its motivation, requirements, architecture, functionality and implementation. For more information on the toolset, I

refer the reader to its technical and user documentation, available from the project website[1]. In the next chapter we turn our attention to our experiences in defining and implementing control panels using the toolset and framework described so far.

[1] http://www.soberit.hut.fi/lucos/

Chapter 6

Case Studies

> I mistrust laboratory methods because what happens in a laboratory is contrieved and dictated. The evidence is manufactured: the cases are what reporters call frame-ups. If the evidence is unexpected or unaccountable it is re-manufactured until it proves what the laboratory controller wants it to prove.
>
> George Bernard Shaw

6.1 Introduction

This chapter presents our experiences in building control panels and applying our toolset in four Finnish high-technology companies. For each case, I first give a short overall view, describing the action research cycles that took place in the case. Then, I describe each case according to three themes: control panel design and implementation, work on measurement foundations, and toolset installation and use. Each case description ends with a discussion of the lessons learned, and the impact the case study had on our constructs.

After the single case studies, I present a cross-case analysis, organized according to the same themes as the single case studies. In addition, I discuss overall lessons learned, success factors for, and obstacles to control panel implementation.

Finally, I discuss the evaluation of the industrial cooperation done by an external consultant, and contrast the results of that evaluation with those we obtained.

6.2 Case 1: SoftCorp

6.2.1 Company Description

SoftCorp was a small, privately owned company providing IT solutions to wholesale businesses. The company was founded in 1982 and employed about 45 people.

The original founder still held the majority of the company shares and worked as President and CEO. SoftCorp was a one-product company whose business revolved around a software product, Trade+, that the company had acquired. The product, a COBOL-based ERP system for wholesalers, was installed at about 250 sites, most of which had their own, tailored version of the program.

The product was fairly complex—it consisted of over 900 separate modules, several of which existed in many versions, due to the company having a policy of tailoring, essentially never turning down customer requests for custom changes.

Product development was organized functionally and unseparable from maintenance. All in all seven persons worked on maintenance and development of the software product. The interface to customers was handled with the help of a customer support function. Customer support handled simple customer problems, training and installation services, as well as screen and report customization which could be done without programming. In the case of a delivery to a new customer or an update that demanded programming, the customer support persons took the role of project manager.

The company fought with two major issues when we started working with them: a chaotic development and maintenance process for the existing product, and a large decline in the sales to new customers. Company management felt that the first problem was due to the lack of defined and followed processes and project management practices. The vice president in charge of product development explained:

> "Our projects are without exception late. We do project plans for larger projects, but they are not updated during the project. It is not uncommon for a four month project to be delivered four months late. Sometimes it is also a deliberate business decision: we have to promise short lead times to get the deal."

The second problem was attributed to the technical outdatedness of the product: most competitors had products with graphical user interfaces running under Windows, and SoftCorp's product was still text-based and worked on dumb terminals or on PCs under a terminal emulator.

6.2.2 Client Infrastructure

The cooperation with SoftCorp was close and active. In addition to the author, one additional researcher worked with the case. On the company side, one person worked full-time on process improvement and measurement development. In addition, the vice president responsible for product development was heavily involved in the cooperation, as was the product development manager. The owner-president participated in meeting and workshops as needed, typically about once a month. The cooperation was intensive, with weekly or biweekly visits.

6.2.3 SoftCorp Action Research Cycles

A summary of the SoftCorp case is shown in Table 6.1 below.

Table 6.1: SoftCorp Case Summary

Cycle theme	Actions taken	Results
1. Familiarization	• Interviews with management and development personnel • Document analysis • APK[a] data analysis	• Increased understanding of current processes and their deficiencies • Decision to improve product maintenance • Decision to develop new process and measures for Trade2000
2. Trade+ development & maintenance (existing product)	• Measurement proposal developed • Toolset installed • EDAMs (csvtool) developed	• Personnel resistance • Toolset installed • EDAMs (csvtool) developed and installed
3. Trade2000 development (new product)	• Incremental process developed and instantiated • Project level measures and control panels defined and implemented • Simple project management practices defined and deployed	• Process and project management practices found to work well • Experiences on defining and implementing visualizations • Data on visualization usage
4. Strategic measurement	• Strategic measures defined using the Balanced Scorecard (Kaplan and Norton, 1992).	• Measures reported to be "good" • Implementation left for the future

[a] in-house developed workflow system

6.2.3.1 Cycle 1: Familiarization

During the first cycle, we familiarized ourselves with the company by interviewing both management and product development personnel, and by studying corporate documentation and existing data. One could basically characterize SoftCorp as a CMM Level 1 company with almost no Level 2 practices in use.

The company had no explicit development strategy, and had no identifiable strategy process. Corporate management had used an external consultant a few times to help them do long-term planning for the company. This was, however, mostly focussed on sales and other financial aspects. At the strategic level, product development was not systematically measured and monitored. The goals related to development were indirect, and mainly related to customer retention and sales growth.

Portfolio management was not explicitly recognized or handled in the organization. Instead, resourcing and other decisions were made ad-hoc as part of normal day-to-day management.

In product development, one could identify two distinct processes: the product maintenance process, which was concerned with maintaining and further developing the Trade+ product, and the Trade2000 new product development process, which the company was in the process of instantiating. In the Trade+ development process, no clear distinction was made between new feature development and bug fixing efforts.

Project management was ad-hoc, with project plans being written only if required by customers. No systematic process for project planning or control existed, and the division of work between developers, customer support and company management was unclear. Software configuration management was not institutionalized, and no tool was used.

The existing measurement infrastructure was based upon the company's APK system, an in-house developed workflow system used by development and customer service to manage their day-to-day work. In addition, spreadsheets and a reporting product, Crystal reports were used. The existing measures were mainly financial in nature and included, in addition to normal corporate financials, customer profitability and tracking of new sales prospects. For the development and customer support functions, only time sheets were used, with hours being logged by work type and customer. The company had, e.g., no quality measures.

6.2.3.2 Cycle 2: Trade+ Development

The first priority set by management was to get more insight into the Trade+ development process. To aid in this, we interviewed the developers working in the process, made a simple project classification, classifying the projects into three categories: new customer projects, in which the software was tailored for and delivered to a new customer; version update projects, in which the installation of an existing customer was updated, and bugfix projects in which critical errors were fixed.

To help increase visibility into development, we studied the existing data in the APK system, and developed a measurement proposal that utilized the existing data. In addition, we made a proposal for software configuration managment that included both procedures and a proposal for a suitable tool.

We installed a preliminary version of the visualization toolset, and developed the csvtool (see Section 5.6) to enable data transfer from APK and the existing spreadsheets into our measurement database.

Finally, we implemented a few pilot visualizations to demonstrate the feasibility of our toolset. At this point in time, the toolset did not yet log access events, so I have no data on the exact number of accesses to various visualizations. The visualizations were not at this point taken into production use, but merely used for demonstration purposes.

In this cycle, we encountered severe resistance from product development personnel regarding both the introduction of measurement and software config-

uration management. Despite efforts to show problem spots and involve personnel, they remained convinced that SCM and measurement would only introduce more overhead to their already stressful work environment. Unfortunately, but perhaps understandably, management adhered to the principle "If it ain't broken, don't fix it.", and froze further efforts to improve the process. The motivation was that Trade+ development, while perhaps broken from a software engineering point of view, was the process that brought in almost all of the company revenue, and therefore was best left in peace. An additional reason was that management wanted to focus the researchers' efforts on helping their new product development effort, Trade2000. This is discussed next.

6.2.3.3 Cycle 3: Trade2000 Development

During the third research cycle, we focused on the company's new product development efforts. The effort, named Trade2000 was related to the need for renewing the product to improve its position in the market, in particular with respect to new customers. Before the program started, a candidate technology had been selected, and a tentative architectural level plan had been developed. The next step, in which we were involved, was to pilot the new technology and architecture, at the same time defining and implementing a simple measurement system for the program.

During this cycle, we defined and implemented an incremental software development process with weekly checkpoints, as well as instituted basic project planning practices and defined and implemented simple measures for monitoring progress. In addition, we defined and implemented the needed tool support, including data collection, analysis, and presentation via control panels. During our cooperation with SoftCorp, the Trade2000 process was followed for two internal release iterations. In these, the measures and data collection system were piloted during the first release, and in normal "production use" during the second release.

Contrary to the problems with the maintenance process, the people working on the new product did not resist measurement, but viewed it constructively. We think that there are two main factors that can help explain this. First of all, the new project was mostly staffed with newly recruited personnel, who viewed their job positively, and had not yet learned the "bad habits" of the company. Second, the project was a high-visibility undertaking getting lots of top management attention, resulting in a "let's show the others how to do things right" mentality in the project.

6.2.3.4 Cycle 4: Strategic Measurement

The strategic measurement system was considered least important by management, and therefore it was decided to implement it last. As described earlier, no explicit strategy for product development existed, and the existing measures at the strategic level were mainly financial in nature. Management did, however, consider strategic measurement interesting enough to warrant its implementation.

The final action research cycle focused on product development strategy and strategic measurement. In this cycle, we worked together with corporate manage-

ment to define measurable strategic goals, and related measures. For this, we used the Balanced Scorecard approach (Kaplan and Norton, 1992). During a one-day workshop, which included a short presentation on the BSC, an initial scorecard was defined.

Though management felt that the development of the scorecard was useful as an exercise, they did not feel that it was worth the effort to implement the measures and add them to the visualization system. Instead, they decided to use the measures infrequently, and do the calculations using the existing systems in the company. Certain measures, e.g., regarding customer satisfaction would require the development of new data collection instruments. No immediate plans to do this were, however, developed. The reason for this was that management felt that a BSC would contain "nice-to-know" information, but that they still had a good grasp of the corporate situation, and it as a management tool in such a small company would not really bring much value.

The main lesson learned from the fourth cycle was that implementation of measurement—as any corporate improvement effort—must tackle a real and acknowledged need. In this case, development of the strategic measures were felt to be useful, but the implementation effort—based, e.g., on the experiences with the Trade2000 projects—was felt to be too big compared to the expected benefit.

Management also thought that the fact that most strategic measures are updated quite infreqently decreases the need for the kind of online support that our toolset provides.

6.2.4 Control Panel Design and Implementation

At SoftCorp, we designed two sets of control panels: a set of pilot visualizations of Trade+ development using existing APK data, a set of panels for the Trade2000 development effort. The control panel development was at this point not systematic, but based upon brainstorming sessions with management, and an analysis of available data that could be most easily visualized.

6.2.4.1 Trade+ Panels

For Trade+ development, we designed one main panel, containing visualizations of shown in Table 6.2. This panel was in pilot use only.

Table 6.2: Initial set of maintenance measures at SoftCorp

Measure	Explanation / Comments
Effort / Cost driver	Plotted as a bar chart for both individuals and groups, e.g., programmers
Hours / Customer	Grouped by cost drivers, plotted as a bar chart
Delivery precision	Plotted as a scatter diagram
Schedule adherance	Plotted as a bar chart grouped according to days late/early

6.2.4.2 Trade2000 Panels

The control panel hierarchy for the new product development effort consisted of one project main panel, from which more detailed panels for specific tasks, effort estimation accuracy, and time reporting statistics were available. The hierarchy of panels is shown in Figure 6.1.

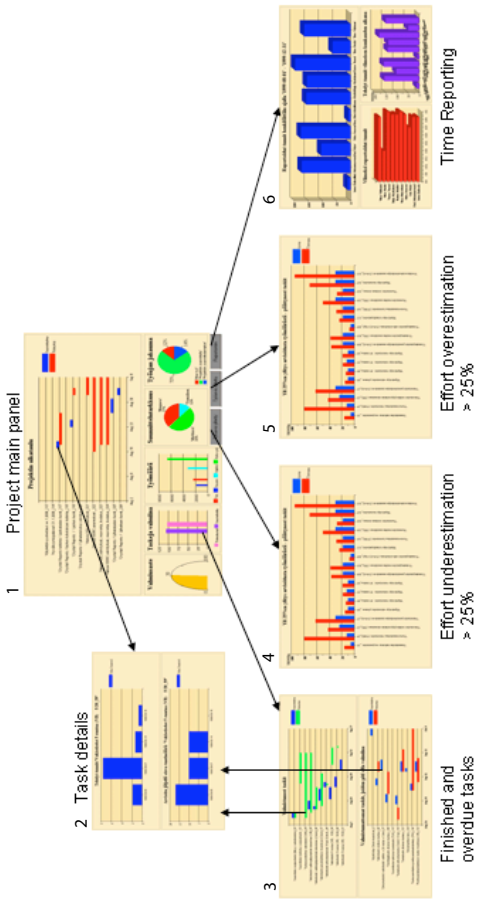


Figure 6.1: The panel hierarchy for Trade2000 development at SoftCorp

6.2.4.3 Additional Panels

In addition, the person responsible for measurement and process improvement at the company developed additional panels both during our pilot project and subsequent to it. During the pilot, he developed panels for a specific project aimed at adapting the Trade+ product for the healthcare business, "HealthProj", as well as for monitoring customer update projects. After the pilot project ended, he defined and implemented panels for monitoring the customer support function.

6.2.5 Toolset Installation and Use

We installed the toolset, and defined data collection procedures and measures for the Trade2000 project. The structure of the measurement system is shown in Figure 6.2. As can be seen in the figure, data was collected from three sources: the software configuration management system, an Excel table in which the developers logged their work hours, and Microsoft Project, which was used for project planning. In addition, data on Trade+ development was transferred from the APK system using an EDAM based upon the csvtool.

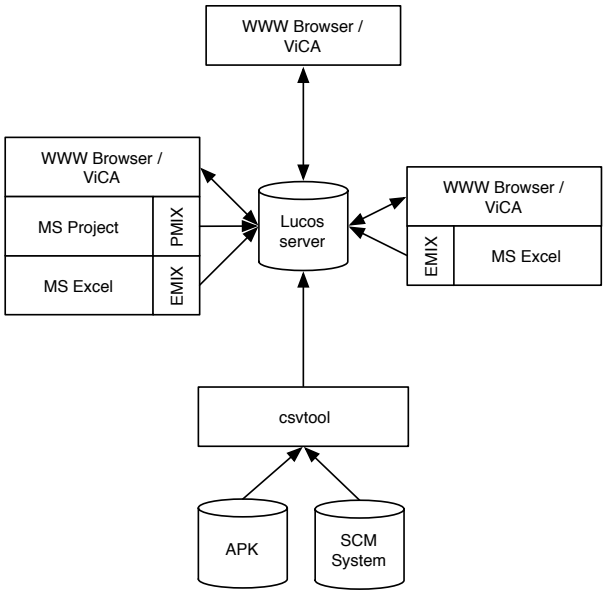


Figure 6.2: SoftCorp measurement infrastructure

ViCA was in real usage at SoftCorp for three weeks, after which the project was cancelled due to the company being acquired.

The short usage of the visualizations means that it is very hard to draw conclusions on the usefulness and hypothesize about what usage profiles would look like if the visualizations had been in real use for a longer time. It is, e.g., entirely

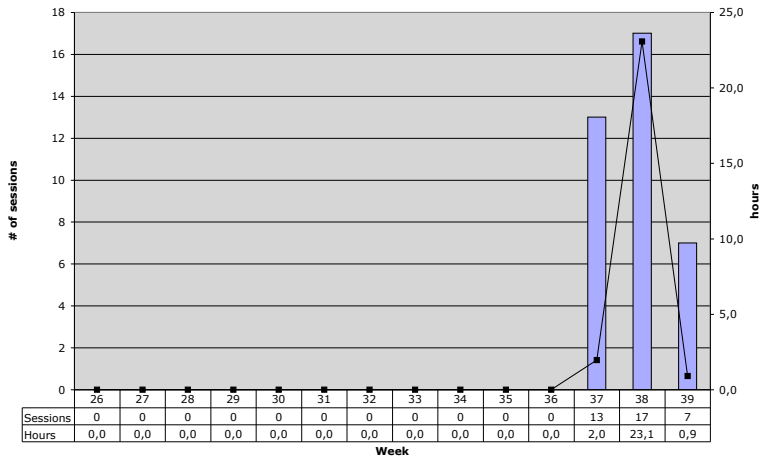
possible that the usage was more active when the toolset was newly installed, for reasons of curiosity.

Despite this, I think that studying the usage statistics available in the ViCA log files—combined with qualitative data from workshops, interviews and meetings with the staff—provide us with some insight into how much, and how visualizations can be used.

We will next look at ViCA usage from four points of view: the developers', management's, project management's, and the measurement system administrator's.

6.2.5.1 Developer Toolset Usage

First, looking at the usage by the developers in the Trade2000 project team, illustrated in Figure 6.3, the first two weeks of deployment shows 13 and 17 sessions respectively, and the third week shows a drop to seven. The session length was on average 9 and 7 minutes during the first and third week, and 80 minutes during the second. The difference was due to a developer keeping the panel open in the background during the second week. However, the important point is that the developers used ViCA on a daily basis during the time it was in use.



	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Sessions	0	0	0	0	0	0	0	0	0	0	0	13	17	7
Hours	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	23,1	0,9

Figure 6.3: Trade2000 developer ViCA Usage at SoftCorp

We will next look at how the developer usage maps to the different panels. I have drawn figures of the usage statistics using the notation shown in Figure 6.4. Using this notation, each panel is depicted as a bar chart with the pilot study timeline on the category axis, and the number of times the panel was accessed

on the y-axis. The gridlines on the y-axis are spaced 5 units apart. The bar on the left of the bar chart visualizes the total number of times the panel has been accessed. Finally, the arrows between panels indicate the direction and number of times navigation has been used to move from one panel to another. Though the toolset was in real use only for the last three weeks, I have opted to show the whole timescale for which we have usage data on the category axis. This way, comparing the usage profiles visually is easier, since they all use the same scale.

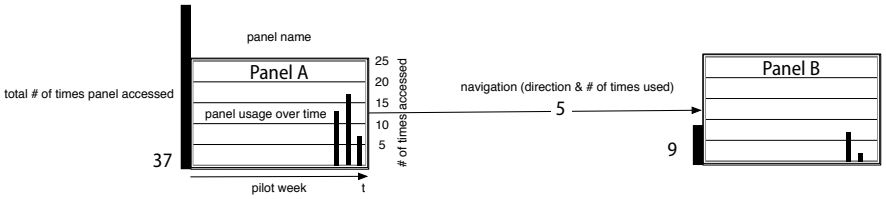


Figure 6.4: Reading the Panel Usage Figures

Figure 6.5 shows the panels accessed both by the Trade2000 developers, and the developers involved in Trade+ development. For the latter, only a specific project, referred to here as HealthProj, was subject to control panel installation.

Looking first at the Trade2000 development team, we can see that they actively looked at the project level panel, Trade2000 project, which was accessed 37 times during the pilot, and which some in the team opened at least daily. From this panel, they navigated nine times to the task info panel, and one single time to the panel showing finished vs. unfinished tasks. When asked, the developers indicated that they navigated to the task info panel to look at some task that looked "strange" in the overall project panel, and that the navigation to the finished vs. unfinished task panel was "just a test".

The panels the Trade2000 team accessed a lot, and that they indicated they were mostly interested in were the Trade2000 reporting panel, which showed the overall time reporting status of the project, and the Employee hours panel, which showed a breakdown of the reported hours on an individual level. The reason for this was made very clear when we talked to them: When asked about the usefulness of the visualizations, the developers indicated that they merely were checking that the data they entered in their time sheets were correctly transferred to the system, and that it showed up correctly in the panels. The reason for this was that they knew management was looking at the panels, and that they therefore wanted to make sure that the information shown was as correct as possible.

Interesting though, is their access to the panels showing estimation accuracy. All estimates were made by the project manager, after discussions with the team. The fact that the team members accessed these panels can be interpreted as indicating their interest in estimation accuracy.

Regarding the overall usefulness of the visualizations, the team indicated that since the project was so small, and all developers were in the same room, they knew the status of the project without looking at the control panels. They did,

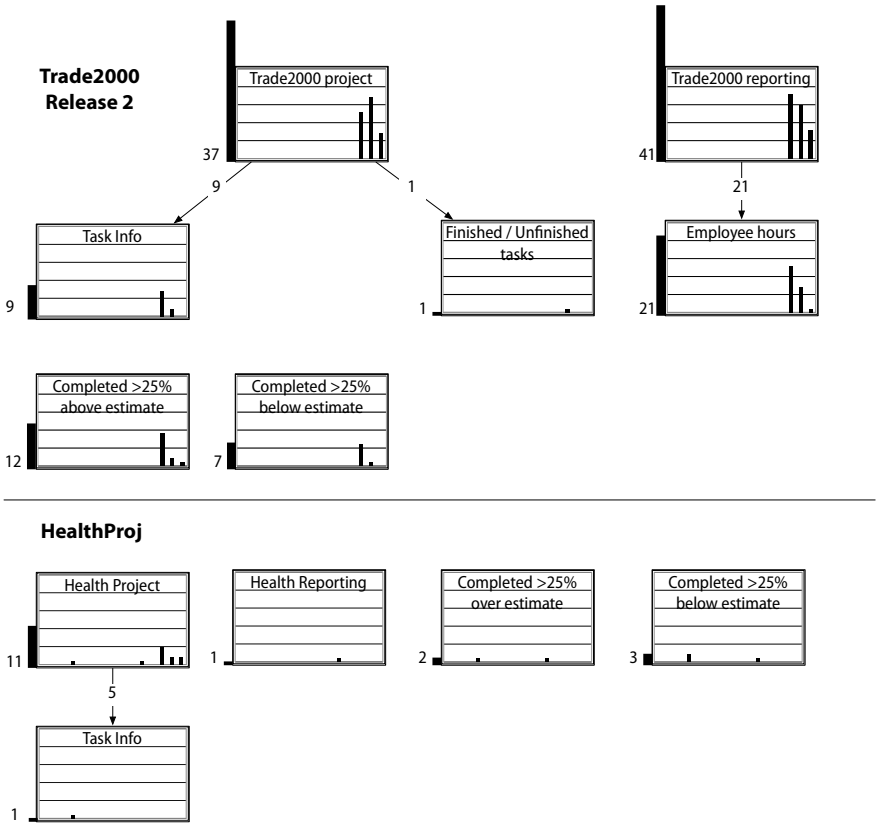


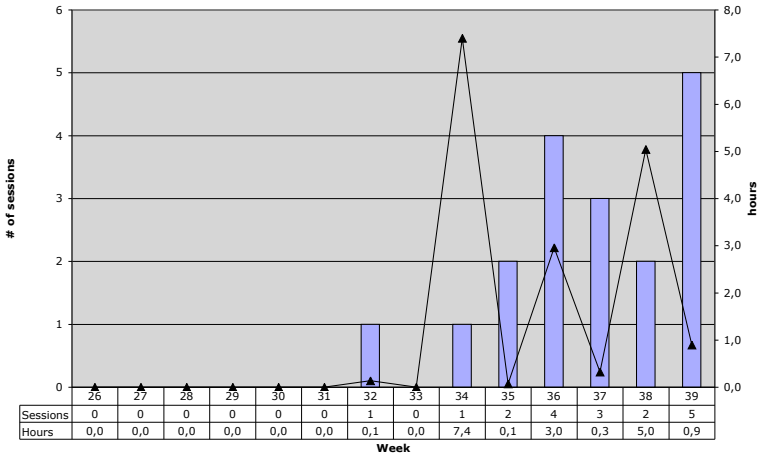
Figure 6.5: Developer ViCA Usage at SoftCorp

however think that the usefulness would increase as the project would grow, and the close contact no longer would be as easy to maintain.

In HealthProj, we had a similar, but less active situation. As the figure shows, the developers accessed the panels only sporadically, with the exception of the main panel, Health Project, which they looked at a few times a week during the last weeks of our co-operation. The control panels and data collection for this project was defined and implemented by SoftCorp without our help. We were also not able to interview the HealthProj personnel, and cannot therefore comment further upon the possible benefits of or the personnel attitude to the control panels. However, remembering their earlier resistance to measurement and control panels, and looking at the quite passive usage, there is little reason to believe that there had been any change of attitude or opinion.

6.2.5.2 Management Usage

The vice president in charge of software development was the person most interested in the deployment of measurement and visualizations. Figure 6.6 shows basic statistics of management ViCA usage.



	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Sessions	0	0	0	0	0	0	1	0	1	2	4	3	2	5
Hours	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	7,4	0,1	3,0	0,3	5,0	0,9

Figure 6.6: Management ViCA Usage at SoftCorp

This was the first time in the company's history in which he had real data on development projects available at his leisure. As the statistics show, he used the visualization system on average three times a week to check on the Trade2000 project. When asked, he indicated that having hard data on a project available made him feel more at ease with the project, since he did not have to live in constant fear about the status of it.

Next, looking at the vice president's use of various panels, shown in Figure 6.7, we can see that he viewed data for all three measured projects.

For the Trade2000 project, he looked at the main project panel 10 times during the pilot, about twice a week. During the second release of the Trade2000 project, in which the system was in production use, he looked at specific task information 24 times. He also studied the status of time reporting and once checked the breakdown of employee time reports. Monitoring the progress of the project, he looked at the "troubled" and "successful" tasks six and five times, respectively.

The vice president told us that he had only looked at the Trade+ panels to check that they worked. For the HealthProj project, he looked at the defined visualizations about once a week, "enough to get a feel for the project". This

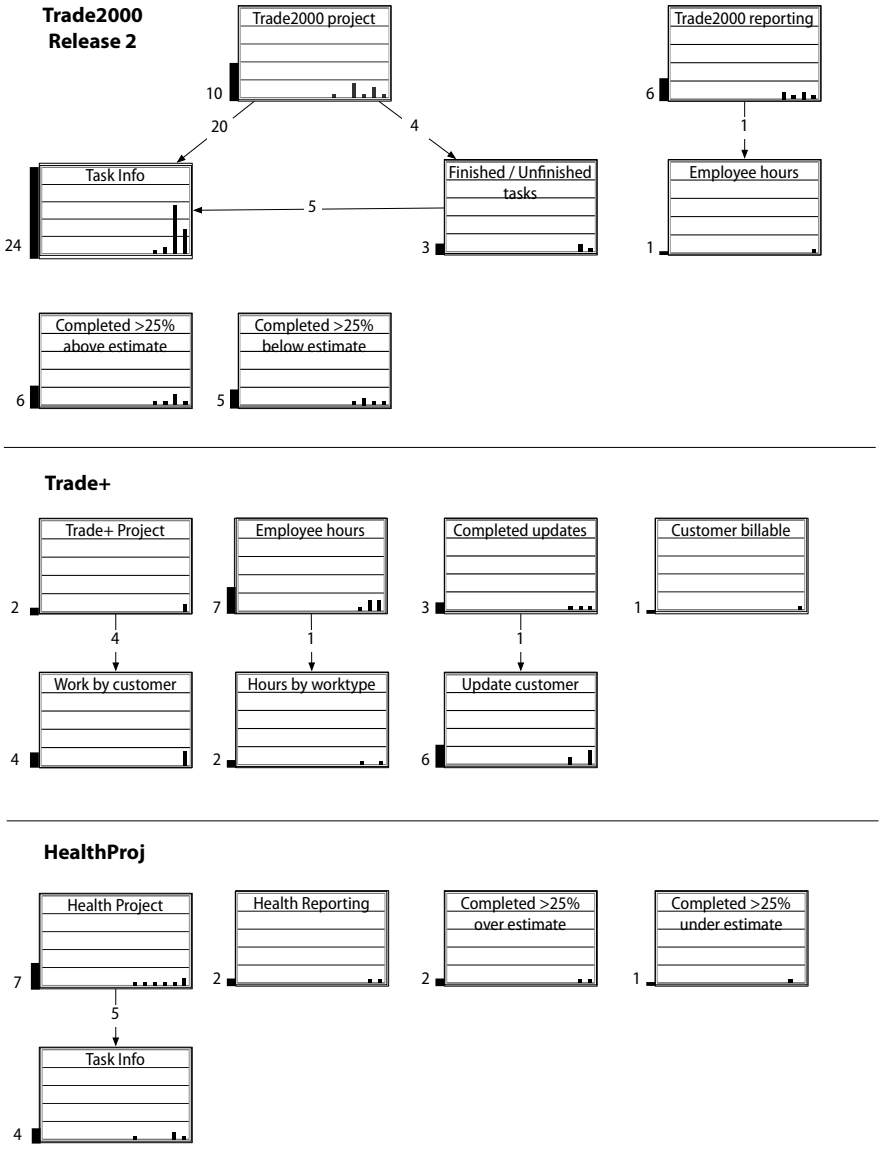


Figure 6.7: Management ViCA Usage at SoftCorp

statement is well supported by the data, which shows a regular weekly access pattern to the main project control panel during the last six weeks of data.

All in all, the vice president expressed his satisfaction with having a working measurement and visualization system that made it easy for him to check up on projects, getting "real data instead of explanations". He also indicated that he used the data to ask targeted questions to project personnel. The measurement system showed its usefulness in a "moment of thruth", at one of the Trade2000 project's steering group meetings[1], as the following anecdote illustrates:

> During a meeting of the project's steering group, the status of the project was inspected using the visualizations provided by ViCA. The system showed that the project was badly behind its planned schedule, and had no chance of delivering all the planned functionality within the time frame planned for the ongoing development cycle.
>
> When confronted with this, the project manager claimed that the information in the panels must be wrong, and that it probably was due to all data not having been entered in the system. The time reporting statistics, also available through the system, did, however indicate that this was not the case. On the project manager's request, the meeting was adjourned while he checked the situation.
>
> When he returned, the project manager, a bit embarrassed, admitted that the visualizations and measures were right. The meeting decided to postpone a major part of the planned functionality to the next development cycle. The old culture of unwarranted optimism followed by disappointment and desperate explanations was broken.

This, simple but still important experience had several effects. First, it was analyzed after the meeting, and especially top management felt that it was a major turning point for the management of product development.

6.2.5.3 Trade2000 Project Manager Usage

The project manager used ViCA only sporadically, as shown in Figure 6.8. The panels that he viewed the most, were the time reporting panel, and the main project panel. The peaks shown in the task info and project panel charts are related to the incident described above. Otherwise, the project manager indicate that he did not find the visualizations, or project planning or monitoring practices very beneficial.

The project manager's lack of interest can perhaps be explained by the fact that the decision to institute project planning practices and measurement was taken by senior management. Thus, it is possible that he did not feel ownership of them, and therefore was not as committed as he could have been.

6.2.5.4 Administrator Usage

The data for the administrative usage allows us to get a preliminary indication of the amount of effort needed to successfully build and test visualization panels.

As the figure shows, the administrator had the visualization application running on his workstation between half- and full time during the fall.

The effort that goes into building charts and panels is interesting. Unfortunately we did not ask the administrator to record the time he spent on this.

[1] At which the researcher was present

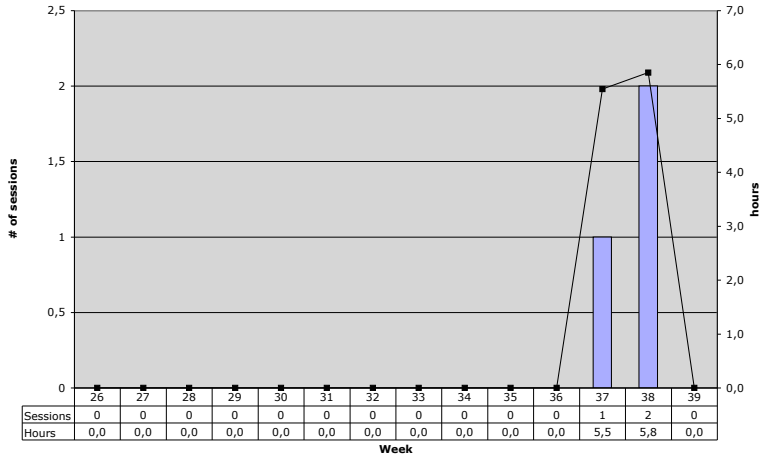


Figure 6.8: Trade2000 Project Manager ViCA Usage at SoftCorp

Instead we have to rely on the log data from ViCA. Each time a user saves a panel or a chart, ViCA logs this. In addition to the event, timestamp and user, ViCA also records the duration for each event. For save events, this duration is the time elapsed between opening the object saved, and the actual save operation. Thus in the following discussion, when discussing "effort", I refer to the time charts and panels have been open in ViCA, and use it as a proxy for real effort. It is very likely that this underestimates the time used to develop the panels and charts, since the administrator did not necessarily keep the visualizations open all the time while working, e.g., on research related to them.

Table 6.3 shows basic statistics on the effort used to create charts and panels. All in all, SoftCorp has 89 charts, grouped into 34 panels.

Chart Creation Effort On average, charts have been saved 2,8 times, and the average total time a chart has been open per session is about six minutes. On average, the time to create a chart is seventeen minutes, ranging from extremely quick saves (3 seconds) to long 114 minutes.

Figure 6.10, shows the total editing time for each chart at SoftCorp, as well as how many times it has been saved. The bars show the editing time in minutes, on the primary y-axis, and the line shows the number of times each chart has been saved on the secondary y-axis. Note that the scale on the primary axis is logarithmic.

The distribution of editing time is close to logarithmic, with a few charts taking up the majority of the editing time. The three top charts stand for 39% of

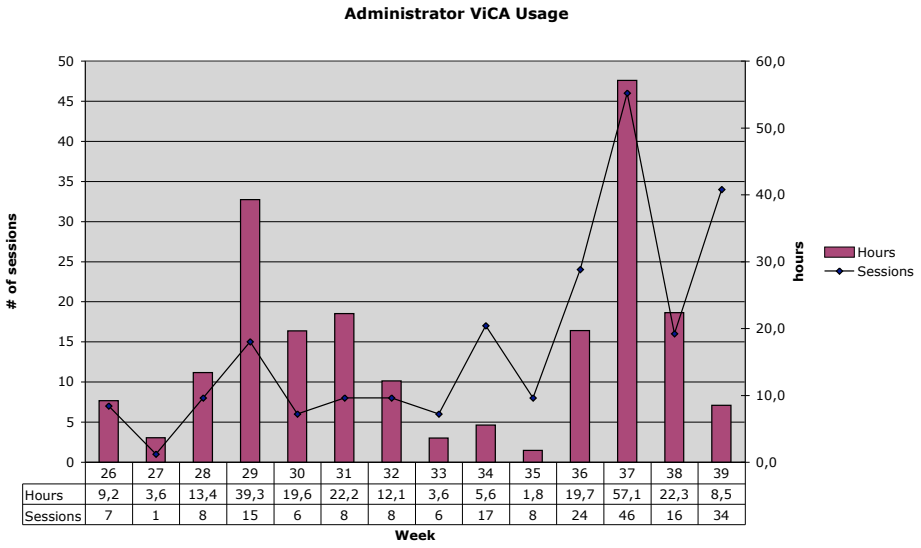


Figure 6.9: Administrator ViCA Usage at SoftCorp

Table 6.3: "Effort" for Chart and Panel Building at SoftCorp

	Panels	Charts
Number of objects	34	89
Minutes, range	0,09–259	0,05–114
Minutes, average/object	81,7	17,0
Minutes, average/session	10,8	6,0
Times saved, range	1-33	1–16
Times saved, avg	7,6	2,8

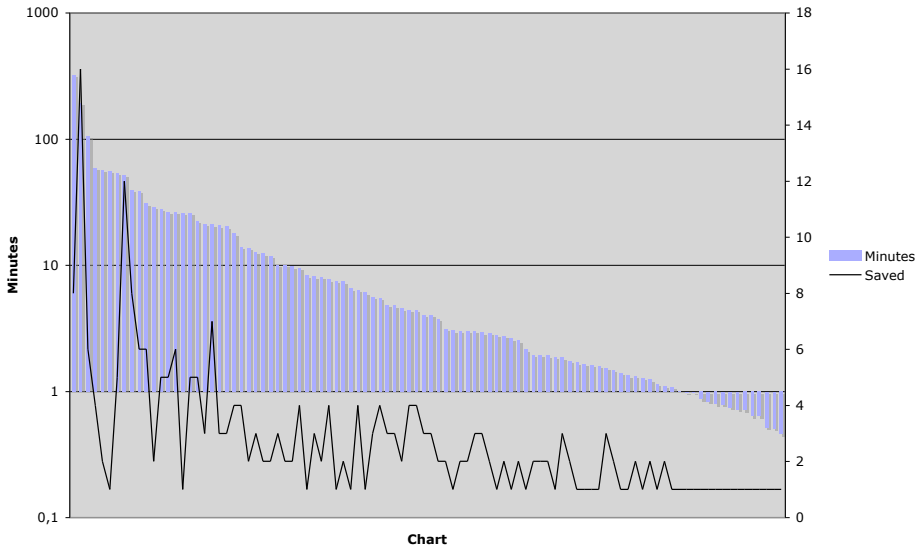


Figure 6.10: Chart Editing Effort and Times at SoftCorp

total editing time, and the top six for 50%. These charts were "new" ones in the sense that their SQL queries have had to be developed from scratch. The ones that have required less time, but still substantially more than the "quickies" are either new charts that are based on very simple queries, or simple modifications of existing charts. The charts with very short editing times are typically direct copies of existing charts. Interestingly, I could find no clear "learning effect"—the development time for new charts of similar difficulty[2] did not require less effort later in the case study. When I talked to the administrator, he confirmed this observation, and noted that basically everything that had required him to develop own SQL queries either early or late in the pilot had required lots of time, patience and effort. He did, however, say that copying existing charts or making small modifications to them became easier as his familiarity with the system grew.

Panel Editing Effort Figure 6.11 shows the editing time and number of saves for the panels at SoftCorp. This distribution is also approximately logarithmic. Looking at the top five panels, they are in order: the HealthProj project panel, a Trade+ customer breakdown, the Trade2000 project panel for the second release, the Trade+ project panel, and the Trade2000 project panel for the first release.

[2] As measured subjectively by my interpretation of the involved SQL queries.

What these panels have in common, is that they are the most complicated ones in the system in the sense that they have the largest number of visualizations. It is interesting that so many of the project panels are here at the top, since they basically are very similar. Thus, it seems that learning has not played a big role regarding the panel development either. When we talked to the administrator, he told us that he typically kept the panels open when struggling with defining the individual charts. This possibly explains most of the data.

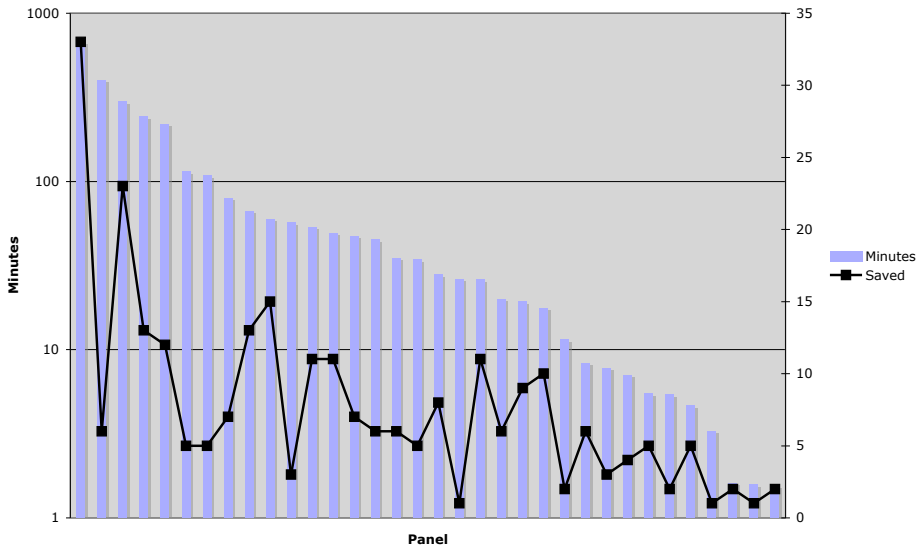


Figure 6.11: Panel Editing Effort and Times at SoftCorp

As for charts, I did not see any learning effect for the development of panels or charts. Instead, more important seemed to be the reuse, in particular of charts. While one would expect similar results for panels, as since the reused panels were later modified, this effect is not detectable in the data. For example, one of the last panels to be developed was the project panel for the HealthProj project, which the administrator developed on his own, without assistance from the researchers. This is the panel that has the largest logged effort (682 minutes), and has been saved the most times (33 saves). The administrator told us that the reason for the large effort was that he had problems with writing and debugging the SQL queries for the charts in the panel.

Not having received formal training in SQL, the hardest part of making the visualizations work was defining the SQL queries needed to define the charts.

On several occasions, we provided SQL training, and helped define and/or debug queries. These problems, which also the other companies suffered from, led us to think about the idea of building visualization libraries, i.e. a set of predefined charts and related queries that only would need to be "mapped" to the measurement database structure of the company.

6.2.6 Lessons Learned

The SoftCorp case was one of our most successful ones when it comes to implementing and testing our ideas in practice. While it is hard to generalize from the limited experiences with designing measures and visualizations, as well as the toolset usage, I think that some of the experiences are interesting also form the point of view of other companies. Understanding the inherent problems of generalization from a short single case study, I nevertheless below present some conclusions that might be interesting more generally than in this single case.

First, the usefulness of a toolset like the one used here from the point of view of single developers seems to be marginal, since the status is easily communicated without tools in small collocated projects. The institutionalization of a simple measurement system was, however, found useful as it made estimation and tracking possible, as well as discussions about project status possible based upon real data instead of opinion.

Second, the project manager also found measurement good, but again, since he was in close contact with the developers on almost a daily basis[3], he also indicated that the measurement toolset was of little direct benefit to him. This observation supports the obvious conclusion that the usefulness of visualization is smaller when it is otherwise easy to communicate status information. Thus, one could hypothesize that the usefulness of tools such as the ones used here increases when communication is harder, e.g., when the project is bigger or the physical distance between project participants grow, such as in distributed or subcontracted projects.

Third, management, which normally was not in daily contact with the project manager or developers experienced the highest subjective benefit of the toolset. The largest benefit was for management, whom did not previously in the history of the company have had real-time (or any whatsoever) project status data available. Again, this more shows the benefits of estimation and measurement than the visualizations. However, the manager indicated that he found the tool useful and intuitive to use.

Institutionalizing measurement in a company with an immature software engineering culture requires a major "cultural" change in a company. In this case, it required a "moment of truth" before the measurement system was accepted.

We learned to appreciate the problems of measurement system implementation. While measurement development was not felt to be a real problem—contrary to what one would expect when reading the software measurement literature—the intricacies of defining working data collection and

[3]The project manager also had other duties.

visualizations made themselves known. While the literature contains tips for building data collection systems, as well as examples of forms for collecting data (Basili, 1984; Park et al., 1996), one would be led to believe that this is "a small matter of implementation", while it actually turned out to be the hardest part. Since we mainly worked with very simple and basic measures, issues like measure validation did not come up.

Finally, we learned that measurement definition is not as straightforward as one could expect when studying available methodologies. In particular, management did not like the GQM, which they considered too hard to understand, an observation made also in another case study (TeleCorp).

In practice, both measures and control panels were developed iteratively, and mainly by drawing the visualizations different stakeholders were interested in.

While, in comparison with our other cases, the SoftCorp case was very successful, it suffers from several limitations. First, the collected data was related only to schedule and effort. Thus, we did not fulfil the criteria of simultaneously measuring several dimensions. We had planned to institute product level measures after the first iteration, but since the project was cancelled, we did not get that far. Second, the usage experience is from such a short time period and gives us so little data that it is hard to draw any strong conclusions regarding measurement or the use of control panels in other settings based upon these experiences.

6.2.7 Impact on constructs

The SoftCorp case impacted our framework and toolset in several ways.

From the framework point of view, having started out as "GQM converts", this case, as well as the ElectroCorp case taught us several problems with the approach, which a later literature survey showed were already known. In particular, we made many observations discussed by Card (1993).

Regarding the toolset, the struggles the company had with defining and debugging the SQL queries made us think about how to ease the development of visualizations for use in the control panels. This led to the introduction of the idea of "visualization libraries", i.e., predefined visualizations that could be delivered together with the tool and "connected" to the data store for the company in question. In addition, we got lots of feedback regarding the usability of the toolset, mainly regarding the administrator interface. Many of these suggestions were incorporated in later versions of the toolset.

6.3 Case 2: ElectroCorp

6.3.1 Company Description

ElectroCorp is a privately owned company in the electronics industry. It was founded roughly 80 years ago, and is one of the largest privately held manufacturers of electrical and electronic products in Finland. It has a global presence with worldwide sales offices. Its operations are located in Finland and the UK. The company has about 450 employees, and its turnover in 1997 was ca 65 M€.

ElectroCorp develops and manufactures a large array of products related to lighting, some of which consists of both hardware and software. Product development is functionally organized, and carried out at both main locations — in Finland and the UK. It is internally organized into three groups, two located in Finland, and one in England. The newly appointed owner-president was worried about serious schedule overruns in NPD. The company also had some historical problems with developing products that the market did not buy. Feeling that there were lots of room for improvement, the president initiated the cooperation between ElectroCorp and us, and delegated the responsibility to his also newly appointed product development manager.

6.3.2 Client Infrastructure

Initially, the owner-president took an active interest in the joint project, in particular during the first action research cycle. He then delegated the project to the product development manager. During the initial phases, the product development manager worked hard on the project. The cooperation was close and active, with weekly or biweekly meetings between researchers and corporate personnel.

During the second cycle, the company hired a person to work full time on improving product development processes and to institute measurement. During this time, the client infrastructure was similar to the SoftCorp case with the exception that meetings with management were held much more seldom. Management also had very little time for the project, since a new corporate strategy, and several other improvement efforts demanded management attention. Thus, most of the work in this cycle was done by the new person, with only sporadic access to management. As discussed later, this turned out to be close to catastrophic from the cooperation point of view.

In the late stage of the project, when simple measures were developed for the Finnish development organization, the cooperation reintensified, with the product development manager again taking an active interest in the project.

6.3.3 ElectroCorp Action Research Cycles

The initial analysis was, again, conducted by interviewing general management, product development management and personnel, as well as people from the main functions interfacing with product development, i.e., sales/marketing and manufacturing. We also studied corporate documentation and available performance data. A summary of our findings is shown Table 6.4.

6.3.3.1 Cycle 1: Familiarization

At the strategic level, top management was just beginning to direct attention to product development, and the development of a product strategy for the various product lines of the organization was under way. The roadmap under formation identified new product lines as well as existing ones, and attempted at tying them

Table 6.4: ElectroCorp Case Summary

Cycle theme	Actions taken	Results
1. Familiarization	• Interviews with management and NPD personnel • Database analysis	• Status report on product development at ElectroCorp • Increased understanding of current processes and their deficiencies • Decision to develop NPD measures
2. Product Development Measurement in Finland	• Stakeholder identification • Stakeholder interviews • Goal definition	• Goal explosion • Prioritization difficult
3. Project Level Measurement and Toolset Piloting in Finland	• Defined basic project management measures • Defined basic project control panel	• Basic time measurement instantiated • Toolset pilot implemented

to a timeline. Management did, however feel that this was hard as the schedule adherence of product development traditionally had been very poor.

Portfolio management turned out to be a serious problem in both Finland and the UK, with the situation being subtly better in Finland. In Finland, the concept of a business plan that identified the market potential for a new product to be developed was being introduced, but did still not work. In England, projects seemed to be started almost randomly, and several respondents indicated that they did not know who decided if and when to start a new project; also the manager in charge recognized this to be a weakness of the current system. The business plan concept introduced in Finland was also applied on a project-by-project basis only: there was no clear oversight of the whole set of projects under way. There were no guidelines for project prioritization and resources were commonly committed to a large number of projects at the same time for some people leading to a "thrashing" situation: they spent lots of time switching between projects. When confronted with this analysis, respondents indicated that the actual situation is much worse—the designers only report the main issues they work on each day. In addition they indicated that there were lots of interrupts from the interfacing organizations, like marketing and manufacturing. The situation was better in Finland, where a separate group was formed to deal with product maintenance and manufacturing support.

The other main issue was a constant feeling of a lack of resources leading to cutting corners in the process. The product development process was documented in the form of a quality system in both England and Finland, however, it

appeared to be a write-only version of the process. Respondents both in Finland and England indicated that the process was adhered to "in the spirit" but existed more for marketing purposes. In England, the interfaces between product development and other departments were very weak; a classical case of functional silos. This lack of internal integration seemed to lead to a lack of customer input in product development, as well as problems in manufacturing.

There was also a clear lack of trust and respect between the manufacturing and development organizations in England. Product development personnel felt the people in manufacturing were not doing their job properly, since they demanded attention for, e.g., finding replacement components due to availability problems. Manufacturing personnel indicated that the product developers did not understand manufacturing; as an example they mentioned that they had over 600 different types of screws for a small number of products. In Finland, the integration with manufacturing worked slightly better; simple component replacements were handled by a component engineer, and manufacturing was involved in product development before finalizing the design.

The functional organization of product development was also visible through the low level of project identification both in Finland and England. There was no clear project ownership in either country. The project management culture could be classified as one based upon scheduling, external deadlines, and hope. The schedules were typically not based on earlier experiences with similar projects, nor did the company separate effort from schedule. This was one probable reason for the over-commitment of resources. At the individual level, product developers complained about a lack of time to do their job properly, and the fact that they had to work on a large number of projects at any one time. Also, interrupts from other departments, such as manufacturing, customer support and marketing were felt to be irritating.

Based on the analysis we made several suggestions to management for improving the state of product development. We also attended a board meeting and presented the results, which were well received. However, unfortunately, no action was taken to improve the situation. When we evaluated the findings together with corporate management, they felt that our analysis gave a good and honest picture of their product development efforts. They indicated that they intended to start improving the processes and in particular portfolio management. For the next cycle, however, they wanted us to start working on measures for the Finnish product development organization.

After the overall analysis, it was decided that we would concentrate our efforts on improving the measures in the Finnish development organization—as a pilot to be rolled out also in the UK if successful. A closer analysis of the product development efforts in Finland showed that the only thing tracked was delivery dates. This, as well as project planning was done on a project-basis, without considering the overall workload of the organization. Thus, the process was trashing, and projects typically never finished on time.

We decided to take two actions: get an overview of the project porfolio, classified into project types, and to build a simple project planning and monitoring system to help the company get improvement started.

6.3.3.2 Cycle 2: Product Development Measurement in Finland

In the second cycle, we started together with corporate management to define organizational goals from which to derive measures for product development. We identified and interviewed stakeholders, and had them fill in goal-measure forms. Unfortunately, this led to an enourmous number of goals, and an even greater number of candidate measures. This experience was similar to what Card (1993) reports, when noting that the GQM approach is non-terminating.

The rational thing to do, of course, is to prioritize the goals and select a small subset of measures. However, we were unable to convince management to attend a workshop in which this could be done. The exercise thus ended as a failure, and I guess ElectroCorp is still counting goals...

While the problem of the explosive growth in the number of goals and candidate measures partly can be attributed to our own inexperience with practical development of measurement systems, it is a known problem with in particular GQM. In addition to Card (1993), other approaches have been developed to help alleviate this problem. For example. Offen and Jeffery (1997) discusses the M^3P approach, in which risk is used to help prioritize and "trim" the goal-measure tree. Since ElectroCorp's strategy and goal state was mostly unclear, it is possible that this explains management's reluctance to participate in goal prioritization.

6.3.3.3 Cycle 3: Project Level Measurement

Following the "failure" of the general product development measurement development, it was decided that the next step was to focus on the project level in one development group in Finland. As the selected group had no existing measures, and performed a large number of simultaneous projects that had no clear plans, we instantiated simple project planning practices, developed a basic time reporting system, and developed a pilot control panel. The effort met with resistance from product development personnel, and since the product development manager did not champion the development, the control panel and measures were discontinued after a short pilot period.

6.3.4 Measurement Foundations

Together with management we defined a set of goals and related measures that could be implemented in the product development teams in Finland. Again, similarly to the SoftCorp case, when it came down to actually implementing the measures, we run into problems.

Management feared that implementing data collection would disturb and upset development personnel that already worked under high-stresss conditions. There are two other factors that I think can help explain the product development

manager's reluctance to actually implement measurement: lack of commitment and organizational politics. His lack of commitment was due to the fact that it was top-management who had brought in the researchers to aid the NPD organization. Since his boss had brought in the project, the product development manager gave lip service to the project, but never actually showed commitment when it came down to actually implementing suggested measures or changes. Since he was quite recently appointed, the product development manager also wanted to look good: at one point he mentioned that he feared that measures might make things visible that he did not want higher level managers to see. Thus, he felt, it was better not to implement measures for NPD.

6.3.5 Control Panel Design and Implementation

At ElectroCorp we developed one single pilot control panel at the project level. Since we were able only to institute some simple time reporting for a few weeks, the panel contained nothing else. Data was collected using Excel sheets, and transferred to MESS using the emix plugin. Technically, the system worked well.

6.3.6 Toolset Installation and Usage

We also installed the toolset at ElectroCorp, as a demonstration. We developed some simple example visualizations using data available in spreadsheets. However, the toolset was not taken into use, though no technical problems surfaced. However, the liaison was extremely critical to having to learn SQL in order to be able to build visualization. This was a major reason for us starting to look at the idea of building "visualization libraries" containing pre-defined visualizations that could simply be "connected" to the corporate measurement database using views.

6.3.7 Lessons Learned

The ElectroCorp case was disappointing and sobering. Most problems that we run into were related to corporate culture and lack of management support. In addition, the liaison, who was the product development manager had an own political agenda into which making the details of what happened in his development organization did not fit. In addition, trust was severely damaged due to the problem of the generation of the enormous number of goals and measure candidates that were never prioritized or selected. Thus, our inexperience in using goal-based measurement probably also played a role. Since the problems were mostly related to measurement in general, and not control panels per se, the case does not refute the idea as such, but indicates the challenges of building control panels for an organization that is not ready and willing to make its processes visible.

6.3.8 Impact on Constructs

The ElectroCorp case was rich in learnings regarding problems of developing and installing measurement systems. From the point of view of our constructs, the

feedback on the problems of SQL queries was perhaps most important, since it made us consider building visualization libraries.

6.4 Case 3: SecuriCorp

6.4.1 Company Description

At SecuriCorp, we worked with the product development organization, which was responsible for developing and maintaining technical products related to the security business. In particular, the organization developed and maintained two major software systems that were market leaders in their field in Finland, with, e.g., most large Finnish corporations as customers.

The product development department had been going through some rough times, and the new manager, who previously had been the owner of a competitor that had been acquired by SecuriCorp considered the situation "critical". The product development organization had a personnel strength of 11 persons, and were currently responsible for maintaining one software system, and developing a next generation system for the company's advanced real-time products. Previously, the department had been responsible also for hardware development, but this had now been outsourced. Due to the downscaling of the in-house development, the manager noted that the best people had left the organization, and he felt he had a difficult competence management challenge. In addition, SecuriCorp was a heavily sales-oriented company. From the point of product development, this meant that features and small changes and related schedules were promised to customers without proper consulting with development. This, in turn, meant that there were an enormous number of simultaneous on-going development efforts that had never really been planned, resourced and scheduled. The problem, the manager explained, was that sales personnel knew the developers, and made them promise features and schedules with no regard for the formal organization.

The development organization had a history of missing schedules; for example, the major upgrade project for the old software system was over one year late when we started working with the company. Consequently, moral was quite low when discussing goals and schedules.

From the point of view of the cooperation, the product development manager expressed his wishes that instantiating control panels and measurement for the organization would help him get the "chaos" under control, and give him real facts that he could use when trying to persuade marketing people to deal with product development in a "professional way".

6.4.2 Client Infrastructure

At SecuriCorp, our main contact person was the product development manager, who was responsible for the cooperation. He worked on the joint efforts only part-time. Other employees were involved as needed, including the CEO and marketing personnel. From the point of view of actually implementing measurement, the lack of resources turned out to be a problem.

6.4.3 SecuriCorp Action Research Cycles

At SecuriCorp, we performed three action research cycles, which are summarized in Table 6.5.

Table 6.5: SecuriCorp Case Summary

Cycle theme	Actions taken	Results
1. Familiarization	• Document analysis • Interviews with management and NPD personnel	• Decision to focus on portfolio management • Presentation of diagnosis to corporate management
2. Portfolio management	• Project classification developed • Compound project plan compiled	• Project classification helped structure the "chaos"
3. Toolset piloting	• Toolset installed and pilot visualizations developed and demoed	• Corporate foundations do not yet support the use of control panels

6.4.3.1 Cycle 1: Familiarization

In the first research cycle, we familiarized ourselves with the organization. The picture that the product development manager had given us in the initial discussions turned out to be largely correct.

The organization had neither a clear strategy for product development, nor any defined process or defined interfaces between sales and development. The organizational culture was all about "bringing money in", which put development in a bad light, since it basically was considered a cost generating organization.

Portfolio management was non-existent, with a multitude of on-going projects leading to a chaotic working environment for the development personnel. We also found that the high number of small, already sold features that needed to be developed quickly, and thus overrun any long-term development efforts, was a very probable reason for the large schedule slippages on the major development projects.

Based upon the results of the first cycle, we decided together with the product development manager and CEO to focus our efforts on portfolio management.

6.4.3.2 Cycle 2: Portfolio Management

We decided together with the product development manager to concentrate on "sorting out the mess" of new product development. In order to do this, we planned to list all existing product development activities, and tried to make a

project classification. Following this, we planned to implement a simple multi-project view of development.

We developed a project classification together with the product development manager, and helped develop an aggregate project plan showing all projects planned and under way in the organization.

6.4.3.3 Cycle 3: Toolset Piloting

In the final action research cycle, we installed and piloted the toolset. Due to lack of resourced from SecuriCorp and the on-going chaos in the organization, we were unable to elicit much help from the organization. Instead, we ourselves developed some example visualizations based upon simple time sheets and the aggregate project plan developed in the portfolio management cycle. We were able to install a running demonstration of these, but the system was never taken into use.

6.4.4 Control Panel Design and Implementation

As discussed above, we implemented only one simple demonstration project control panel that was not taken into production use.

6.4.5 Measurement Foundations

SecuriCorp had no working measurement system for product development. In addition to helping the product development manager to build an aggregate project plan, we also helped design a simple time recording sheet.

6.4.6 Toolset Installation and Usage

At SecuriCorp, we installed the toolset using a setup similar to the one at SoftCorp. However, in this case we were only able to develop some simple demonstration visualizations of the project portfolio, and some time reporting graphs. The measures were not taken into production use. Again, we did not run into any technical problems with the tool.

6.4.7 Lessons Learned

The main value of the SecuriCorp case was that it provided us with an opportunity to do a project classification. We also learned, again, the importance of management support, and of involving the practitioners.

The product development manager indicated that he felt that the project classification was beneficial and that it would make it easier for him both to plan and monitor the status of his development organization, as well as to communicate with theimprot CEO and the sales organization.

6.4.8 Impact on Constructs

Since we did not get much practical experiences in the SecuriCorp case, it had no impact on our constructs, with the exception that it confirmed the need for visualization libraries.

6.5 Case 4: TeleCorp

6.5.1 Company Description

TeleCorp is a product development site of a large multinational communications technology company. The site, which is situated in Finland, employs around 1000 people, of which around 600 work in product development. We worked with a unit within the site that developed embedded software for complex communications systems. At the time, there were ca 130 people working in the unit. In the following, TeleCorp refers to this one unit unless otherwise stated.

Product development at TeleCorp is organized as a matrix; line management is responsible for hiring, training of personnel, as well as for allocating them to projects. Project management is responsible for achieving the project goals utilizing the resources allocated by the line organization. At any one time, TeleCorp had one to three simultaneous active projects. A typical project lasts from nine to twenty months, and there is a constant need to shorten cycle times. TeleCorp has a fairly old process culture; the corporate product development process is defined in detail, and has been used for over 10 years. The process was not only accepted by all personnel; it is seamlessly integrated into how things were performed in practice. The process is fairly stable, and the organization has a good record on meeting project budgets and deadlines.

The divisional level at TeleCorp had a few years earlier received the Finnish quality award, which is awarded according to the Malcolm Balridge criteria, for its excellence in product development. TeleCorp also has a long history of measurement, both at the global corporate level, and at the national and unit levels.

Lots of measurement data was collected into various databases, and for various uses. Despite both an old process culture, TeleCorp had identified problems both in applying its "one-size fits all" process to all kinds of projects; and in the effective and efficient usage of measurement data. The first problem had already resulted in developing an alternative process that was more iterative in nature than the old, sequential process. Our case study, however, focused on the second problem, measurement. In particular, there were problems in both communicating and using measurement data, and a problem with a measurement system to which new measures were constantly added without removing old ones. This led management to believe that not all existing measures were relevant in the current situation.

6.5.2 Client Infrastructure

In the TeleCorp case, there was initially two persons working part-time on the project, and later one full-time person was added. In addition, project office, quality organization and other personnel participated in joint activities and workshops as needed. The commitment of TeleCorp was thus excellent. However, when the project champion, who was the division manager, moved to head another TeleCorp development unit, commitment dropped, as his successor did not share his enthusiasm and interest in organizational measurement.

6.5.3 TeleCorp Action Research Cycles

A summary of the TeleCorp case is displayed in Table 6.6.

Table 6.6: TeleCorp Case Summary

Cycle theme	Actions taken	Results
1. Familiarization	• Document analysis • Personnel interviews	• Complex, fragmented measurement system • Need to simplify, study relation of current measures to organizational goals
2. Measurement system study	• Document analysis • Workshops with division and project management	• Measures identified • Goal-measure mapping done at divisional and project levels
3. Toolset piloting	• Toolset installed • Pilot visualizations developed and demonstrated	• Toolset development suggestions • Pilot visualization works • Difficulties getting access to and understand the corporate databases • Need for caching query results

6.5.3.1 Cycle 1: Familiarization

In the initial cycle, we familiarized ourselves with the measurement programs and infrastructure at TeleCorp. As part of a large international group, the measurement system had during the years become complex and fragmented. Many measures were collected as part of corporate wide policies, and others were locally defined. There was no clear understanding of the measures and their links to the explicit goals of the unit under study. As a result, it was decided to study the measurement system in more detail. This was the theme of the second research cycle.

We also performed a study of exisiting measurement infrastructure, and found that measures were collected into seven different databases, some of which the local organization had no control over, and only limited access to. The databases were from different vendors, and we were not able to get data dictionaries for all of them. Due to this, it was decided to start the tool piloting using only local databases that were easily accessible. Toolset piloting was the theme of research cycle 3.

6.5.3.2 Cycle 2: Measurement System Study

In the second research cycle we studied the existing measurement system by more detailed document study, and by trying to identify links between measures and organizational goals. This was done in workshops with company personnel, and is discussed in more depth below.

6.5.3.3 Cycle 3: Toolset Piloting

In the third research cycle, we piloted the toolset. Due to the heterogeneous and distributed nature of the corporate measurement programs and measurement databases, the organization had communicated a clear need for control panels that could help integrate different measures into simple visual displays. In the prevailing situation, different systems generated a variety of measurement reports, but there was no consolidation or integration activities. In this cycle, we installed the toolset and implemented a few pilot visualizations at the project level.

6.5.4 Control Panel Design and Implementation

In the TeleCorp case, control panels were not systematically defined. Instead, simple visualizations that could be build based upon existing data were developed by the corporate personnel on their own. The idea was to first pilot connections to the most important databases, and when that worked, start building control panels. Unfortunately, we did not get to that stage.

6.5.5 Measurement Foundations

The work on measurement foundations at TeleCorp focused on analysing the existing measurement system. The organization explicitly stated that it was not interested in new measures, since it was felt that the current number of measures was more than adequate.

In an effort to better understand which measures were important and useful for the organization, we performed a goal-measure linking exercise, in which we together with corporate personnel tried to link existing measures to explicit corporate goals. As the measurement programs had grown over the years, based on various efforts, there was no clear view of the whole and its relation to the actual goals of the organization.

We started the goal-measure mapping by interviewing the division's quality manager about the existing measurement system. The measurement system was

newly created, and based upon the Balanced Scorecard (Kaplan and Norton, 1992). The measures were grouped into four dimensions: customer, people, process, and product. Since, according to the Balanced Scorecard, the measures should be derived from the company strategy (Kaplan and Norton, 1993), we expected the goal-measure mapping to be easy to do; ideally, it should already exist and be documented. It turned out, however, that the measures in the balanced scorecard were derived in management workshops by brainstorming and discussions , without explicitly linking them to objectives of the organization. The balanced scorecard was used only as a template guiding the discussions and giving a format to the final outcome. Consequently, we had to do the mapping afterwards. We initially tried a "reverse-GQM"[4] trying to apply the Goal-Question-Metric paradigm (Basili and Rombach, 1988) in the opposite direction, but it turned out to be artificial to derive the questions this way. We therefore decided to leave out the question-level, and to do a direct goal-measure mapping.

The mapping was done in a workshop with the quality manager of the division, acting as a representative for the management team that had developed the measures. It would have been better to include the whole team, but unfortunately we did not have that possibility; also, the division manager heading the effort had transferred to another organization within TeleCorp. Before the workshop, we had prepared small paper notes, one for each measure, and one for each goal. The measures included both the key performance indicators from the balanced scorecard, and other institutionalized measures, which we called performance indicators. The measures were taken from the official documentation found in the division's quality system. The goals were taken partly from the quality system, partly from the official documented strategy. The process of finding goals and measures was easy since they were well documented. All in all we found 18 goals and 20 measures. The actual mapping was done in two phases, using a "wall-technique". We first simply asked the quality manager to take one goal or measure, stick it to the wall, and then find the corresponding measure(s) or goal(s) and stick it (them) next to the first paper. This turned out to be almost impossibly hard; we therefore first asked the quality manager to cluster goals and measures together, making the amount of goals smaller. When this was finished, he was asked to name the clusters. Finally, he was asked to link the measures with the goal clusters, and when possible with a specific goal.

The results were interesting and eye-opening to the quality manager: of the 18 goals, only four had strong links to measures, and three others had weak links. Of the 22 measures, four had strong links to some goal, and two had weak links, giving a total of 18 measures that were not linked to any explicit goal of the division. In addition to the linking of goals and measures, we asked the quality manager to give an overall "goodness" rating for each measure on a scale from one to five. He was furthermore asked to prioritize the goals, but since the goals were not prioritized when the strategy was defined, we decided to leave the goals unprioritized. Finally, he was asked to group the measures into three groups based upon their

[4] The idea to try to do a GQM tree post fact was our own, and is not part of the GQM approach.

importance as a means of control. The groups were: a) strong/directly controlling, b) weak/indirectly controlling, and c) no control function.

After the workshop, the researchers documented the results in a matrix, a modified version of which is shown in Figure 6.12, which was given to the quality manager for comments and corrections.

In the figure, the rows contain the corporate goals, listed in order of priority. The columns list the measures, in order of "goodness", as evaluated by the workshop participants. A yellow box indicates that there was a weak linkage between the goal and the measure; a green box indicates a strong link, and a white box indicates that no link was identified. In a well designed measurement system, one would expect each important goal to be linked to a measure. In the case of a one-to-one linkage in which the goodness or importance of the measures perfectly matches that of the goals, the matrix would contain a diagonal line of green boxes from the upper left to the lower right corner.

In additional to the divisional level, we performed goal-measure mappings at the unit level, and for two projects. The observations and main recommendations for improvement made by the personnel are listed in Table 6.7.

The linking of existing goals and measures proved to be both much tougher and more revealing than we had initially expected. First, even finding the measures and goals to map was challenging; they had to be dug out of a large set of various documentation. Second, performing the mappings in a workshop setting proved to be valuable also for the participants — ideas and assumptions came out clearly perhaps for the first time; lots of improvement suggestions came out, and many were acted upon by the organization.

6.5.6 Toolset Installation and Use

The measurement toolset was installed at TeleCorp at two sites; in two separate European countries. In Finland, the toolset was connected to several databases, including the inspection, time reporting, the corporate measurement database, the modification handling system, and the test progress support system. The main expected benefit was to be able to view up-to-date data from several sources in one single panel. Previously managers and personnel had had to rely on various reports, typically in paper format.

Toolset deployment, even for testing purposes, was made more difficult by the heterogeneous nature of, and geographical distribution of the measurement databases. The heterogeneous nature of the databases meant that connection modules for different databases had to be written individually, since the databases were from different vendors. The geographical and related organizational distribution limited the possibilities to get documentation of, and access to remote databases. As such, the large number of connections that was made possible was a big success.

When working with remote databases, some of which responded to queries very slowly, we decided to start caching query results in order to have control panels update more rapidly. The caching mechanism did not only store the results

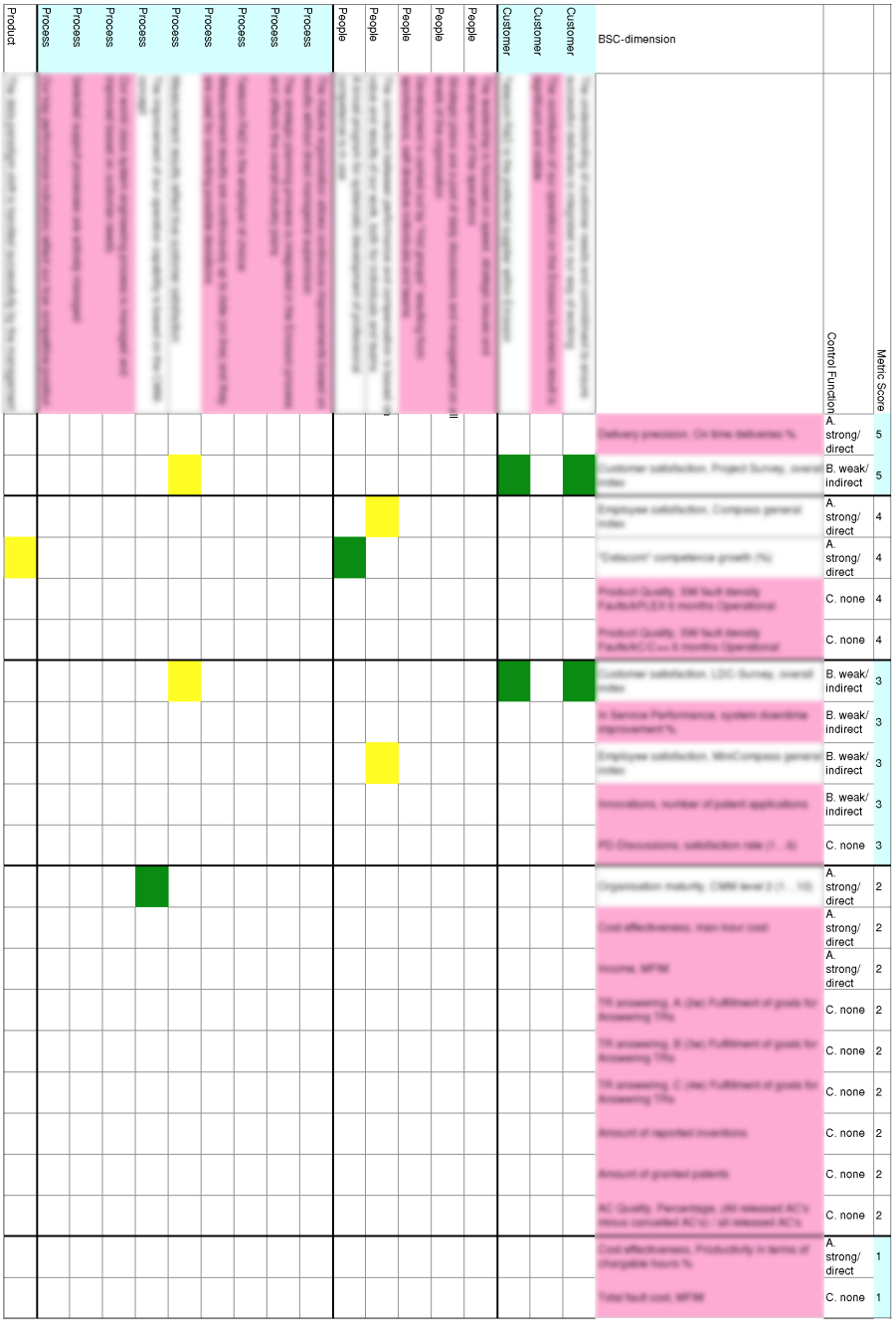


Figure 6.12: The Goal-Measure Mapping at the Divisional Level

Table 6.7: Main results of TeleCorp Goal-Measure Mapping

Object of study	*Findings*	*Comments and proposals*
Division	• Measures mostly operational, not strategic • Strategy not clear enough to make measurable goals • Implicit strategic goals reflecting the corporate culture were not explicit in the documentation	• Improve the strategy process and documentation • Clarify the role of the division versus the operational units
Development Unit	• Some goals not made explicit • Many action type goals • No competence measures • Almost half of the measures are used only for monitoring, and they therefore lack target values • Measurement definition process only indirectly driven by goals • Some target values are goals in themselves	• Make link between goals and measures clearer • Use levels in the scorecard • Make hidden assumptions explicit — discuss and agree • Make explicit action type goals and measures only for major actions • Keep number of goals small
Project α	• Basics in order — the most important goals were measured • Many goals were not measured • Only 2 good measures (as assessed by the project manager) found	• Improve goal-setting • Make higher-level goals more concrete • Reduce number of goals • Follow-up[a] vs. measurement • Reconsider the use of low-scored measures
Subproject α_1	• Only most important goals connected to measures • Some measures not within the tester's scope	• Improve goal-setting and documentation • Make clear who monitors which goals • Test reports have lots of measures — which of them are really needed?

[a] Monitoring vs. control?

of old queries, but actively polled the remote databases. Therefore, we referred to it as "proactive caching", the details of which has been described elsewhere (Risku and Rinta-Aho, 1999).

Unfortunately, however, despite working database connections, we did not during the project get to defining and implementing control panels. The main reason for this was personnel changes—the divisional manager who was the driving champion of the project transferred to another unit. His successor was not at all interested, and consequently lower levels of the organization also showed less commitment. In addition, the organization expressed concern over the fact that our toolset was a research prototype, and therefore unsuitable for production use.

Instead, we ended up with only a handful of working pilot visualizations, one of which is shown in Figure 6.13.

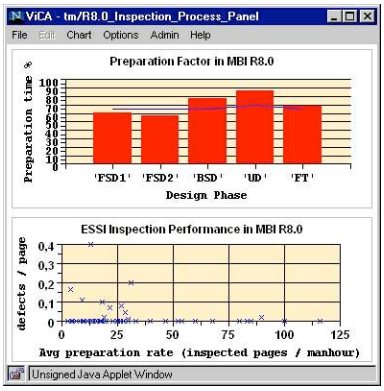


Figure 6.13: An Example Panel showing Inspection Data

6.5.7 Lessons Learned

Installing the toolset into an organization with lots of existing measures and measurement databases proved to be a valuable experience that gave lots of input for the further development of the tools. The actual building of working visualizations from several databases proved to be more involved than we had expected. Also, unexpected problems with the database of an external vendor slowed the visualization implementation down. It was therefore a bit disappointing that we did not get any real data on the use of the visualization tool. This was especially unfortunate in this case, since TeleCorp provided a fairly stable environment into which the tool could be installed; thus it would have been easier to try to get at the possible benefits of the introduction of visualizations in this case than in the others, in which other large changes were simultaneously made.

Also in this case we learned the value of management commitment, and of having a champion for such an undertaking. The complexity of measurement programs and databases in a large organization was enlightning.

6.5.8 Impact on Constructs

The main impact the TeleCorp case had on our constructs was the addition of "proactive caching" (Risku and Rinta-Aho, 1999) to our toolset. In addition, we had to add support for an additional database, but this can be seen only as a small technical issue that, however, indicates the need for flexibility in database support, since organizations might resist investing in database technologies from several vendors.

The original tool architecture was based on the idea of replicating measurement data to our centralized server. This approach was unfeasible at TeleCorp because of the prohibitive amount of data in some corporate-wide databases that we needed to integrate into the system. Therefore, we changed the system to allow for running queries to other databases, to avoid having to replicate the data. This lead to a new problem: some queries to these large databases took several minutes to run, leading to unacceptably long times for opening visualization panels. We therefore introduced caching of queries with a mechanism we called proactive caching , which aimed at providing the newest available data even though we did not have any mechanism for identifying when data had changed in the remote databases. We also had unexpected technical problems with one external vendor's database that delayed the implementation. Partly due to this, we were not able to develop visualization panels for production use during this pilot. The company, however, continued the efforts to implement and use the toolset after our pilot project ended. An example of a panel developed during the pilot and depicting inspection related measures is shown in Figure 6.13.

6.6 Cross-case Analysis

In this section, I draw together the lessons learned in the case studies. I have structured the discussion into three themes: lessons learned, success factors, and obstacles. The lessons learned are further structured into lessons regarding the actual design and implementation of control panels, and lessons regarding work on the measurement foundations necessary Finally, I comment upon how the case studies impacted the constructs, and thus how I think that they in their present form can help overcome some of the challenges that we encountered. Table Table 6.8 shows a short summary of the case studies.

6.6.1 General Lessons Learned

The perhaps most important overall lesson we learnt was that implementing a vision like the one discussed in this thesis, while intuitively appealing and simple, is doable, but not at all simple in practice. In particular, the importance of the "foundations" behind the panels: processes, measurable goals, and working measurement is emphasized. Getting this foundation working to the level that meaningful control panels can be developed is a difficult task that can take several years, especially if aiming at organization-wide control panel development and deployment.

Table 6.8: Summary of Empirical Studies

Study	Objectives	Results/Contributions
Case 1: SoftCorp	• Explore the control panel idea in an SME setting • Build and test control panels at the project, and strategic levels	• Analysis framework • Approach feasible also in small companies/projects • Data and experiences on toolset deployment • Data and experiences on small-scale control panel usage
Case 2: ElectroCorp	• Analyse the product development process • Develop and install visual control panels for product development	• Goal and measure number explosion when using goal-based measurement • Experiences with corporate resistance, politics and fear of measurement • Understood need for visualization libraries
Case 3: SecuriCorp	• Build and install control panels and measurement system for a small software development organization within a large company	• Experience with project classification and portfolio management • Toolset piloting experiences • Understood need for visualization libraries
Case 4: TeleCorp	• Analysis of existing measurement system • Piloting of toolset using existing measurement infrastructure	• Goal-measure mappings • Toolset features: remote queries and proactive caching • Toolset piloting experiences

6.6.2 Control Panel Design

As is clear from the case studies, we did not design and implement control panels to the degree we had expected. The development was also quite unsystematic and much based upon trial and error. We found it easy to sell the idea of control panels, to identify stakeholders, and to have them sketch the things they were interested in seeing. However, we did not get enough experience in systematically defining and implementing control panels to be able to give authoritative guidelines for it.

6.6.3 Control Panel Foundations

Building the necessary foundations for the control panels turned out to be much more demanding and time-consuming than we had expected.

In the ElectroCorp case, we were unable, despite several attempts, to get clearly specified strategic goals that could have provided a foundation for both

strategic control panels and derivation of lower level goals. Instead, we experienced "goal-explosion" (Card, 1993), a known problem with goal-based measurement. On the other hand, the goal-measure analysis at TeleCorp was considered valuable by the organization[5]

At SoftCorp and SecuriCorp, we found project classification valuable. It helped organizations in three distinct ways: by making it easier to grasp the overall situation in the project portfolio, by making it easier to allocate and share resources between projects, and by making project progress monitoring easier. We found no single way or dimensions that were useful in all organizations, but did the classifications on a case-by-case basis.

In order to succeed in measurement system implementation (which proved to be much harder than measurement system definition), it is important to create a positive and constructive athmosphere. This helps improve the reliability of the data, improves the use of the measurement results, as well as helps tackle resistance to change. We found that the most crucial phase is the first usage experiences. Having to deal with facts instead of opinions can be difficult to people not used to it. Measurement data might expose surprising facts that some people have invested a lot in denying.

We also found that making data collection work is no easy task. Some organizations collect lots of data, but the control aspect (i.e., target values, either time-based or not) are lacking. Implementing working data collection procedures can be quite cumbersome and demand lots of time and effort. It can be challenging to motivate product developers to collect data, since they might not easily understand the benefit from their own point of view. We found it useful to gradually implement data collection, and to start with simple solutions, such as using Excel spreadsheets for collecting some very basic data.

6.6.4 Toolset

Our experiences with the toolset was on the whole positive. We found ViCA to be a good tool for motivating the personnel to enter data, since they immediately could view the data and its use in the tool. ViCA was found to be easy to use—at least from the perspective of basic users. Implementing the visualizations did, however, demand an understanding of both SQL and the internal corporate measurement databases. All case organizations, with the exception of TeleCorp, which had dedicated technical support personnel for measurement, deemed this the biggest problem with the toolset.

6.6.5 Success Factors and Obstacles

In this section I try to identify success factors for and obstacles to control panel development. Since the number of cases studies is limited, and much work went

[5]The goal explosion we encountered in the ElectroCorp case was much due to events out of our control: we several times asked management to prioritize the goals to help select a subset, but were unable to get that prioritization. Management did also not take the time to work jointly with us, e.g., in a workshop on prioritizing goals.

into measurement foundations, the discussion here must be considered tentative, not authoritative.

Table 6.9 show the success factors we identified, and how important they were to the different cases. As success factors, I have included factors that I, based upon our experiences, consider necessary, but not sufficient for succeeding in building control panels. In the table, a plus sign (+) indicates the the factor affected the case positively, and a minus sign (-) that it (or the lack of it) contributed in a negative way.

Table 6.9: Success factors

Obstacles	Soft-Corp	Electro-Corp	Securi-Corp	Tele-Corp
Management commitment at all levels	+/-	-	-	-
Resourcing	+	+	-	+
Selling and involving personnel	-/+	-	-	+
Trusting relationship with researcher(s)	+	-	+	+

The most salient success factor was, perhaps not surprisingly, management commitment. Strong management commitment both from the CEO and the VP responsible for the joint project at SoftCorp helped us solve many problems both with resourcing and personnel motivation. The case is particularly interesting in the sense that strong management commitment regarding Trade2000 development helped drive the effort forward, and the lack of it for Trade+ development resulted in failure for that effort. At ElectroCorp, the CEO was committed, but delegated the project to the NPD manager who was committed in words but not in deeds. At SecuriCorp, the situation was the opposite: the NPD manager was strongly committed, but his CEO did not show real commitment, but respected his subordinate's wish to participate in the joint project. At TeleCorp, initial commitment was strong, but faded when the division's head moved to another subsidiary. In all these cases, I think that the failures or very partial successes can be tied to the lack of sustained management commitment.

In addition to management commitment, the development of control panels and related measurement foundations is a substantial undertaking, requiring adequate resourcing from the company's side. While the effort is smaller if targeted properly and if the organization has a working measurement foundation, it still requires a clear commitment of resources to succeed.

The need to sell and resell the measurement effort, and to involve the personnel is crucial to success, as has been noticed in the literature on software measurement program implementation. This is particularly important in companies in which the measurement culture is new or non-existent.

Finally, having a high degree of trust between the researcher/consultant is important. In particular in the ElectroCorp case, lack of trust was salient.

6.6.5.1 Obstacles

There are several organizational factors that must be dealt with in order to succeed in implementing tool supported measurement programs. Table 6.10 shows the most salient ones. These include fear of measurement, organizational change, organizational inertia, organizational politics, organizational culture, resistance to change, and researcher related factors.

Table 6.10: Obstacles

Obstacle	Soft-Corp	Electro-Corp	Securi-Corp	Tele-Corp
Organizational change	+	+	+	+
Organizational inertia	+	-	+	+
Organizational politics	-	+	+	+
Organizational culture	-	+	+	-
Inexperience of researcher(s) regarding substance	-	+	-	-
Inexperience of researcher(s) regarding collaboration with industry	-	+	+	+

Implementing measurement programs is resource and time-consuming, as discussed in the previous chapter, and it is easy to underestimate both the calendar time and effort needed before the measurement system is up and running. During the time we worked with our cases, ElectroCorp faced at least three major reorganizations, SoftCorp was sold and subsequently reorganized. Management at TeleCorp did also change during the case, severely decreasing the commitment of the organization. Reorganizations and personnel changes are a fact in corporate life, but this can lead to significant extra work. Each organizational change can, e.g. require the reselling of an improvement effort.

Organizations on the other hand also have inertia, making change hard. In particular in large organizations, like TeleCorp, seemingly simple things like getting access to databases can be very difficult and time-consuming.

Organizational politics can also stifle measurement program implementation. In the ElectroCorp case, the NPD manager did not want to implement measurement because he seemed to fear giving his superiors additional visibility into his department. In the SecuriCorp case, top management was very sales oriented, and therefore not very interested in developing measures and processes for product development, since they felt that it anyhow was too slow to react and that implementing more measures and processes could give the already problematic NPD manager more weapons to fight with against their wishes and whims.

The success of measurement programs is also affected by the corporate culture. For example, at TeleCorp, the process culture was over 10 years old and could be considered fairly mature. Measurement was already a part of the daily life, and

little resistance was felt. In the less mature companies, like ElectroCorp, SecuriCorp and SoftCorp, measurement was both feared and openly critisized.

In action research, the researcher is intimately involved in the work carried out to improve the client's situation. Thus, the researcher's knowledge, "presentation of self" and other researcher related factors can affect the outcome of the relationship, similarly to what has been discussed regarding qualitative evaluation research. (Patton, 1990).

The fact that we as researchers got our first experiences with measurement system implementation during this project meant that we, at least in the beginning, were as inexperienced as the people in the case companies in defining and implementing measurement. Due to this, we made several mistakes that in, e.g., the ElectroCorp case led to a loss of trust[6].

We also lacked training in research methodology, a fact that is discussed in Section 7.4, and not further considered here, since it possibly did not significantly affect the outcome of the industrial collaboration.

Summarizing the discussion of success factors for, and obstacles to control panels, they are well in line with what is known for development and installation of measurement program in general (Goldenson et al., 1999; Berry and Jeffery, 2000). This makes it no doubt a challenging undertaking, the success factors for which are much related to organizational change and commitment of adequate resources. All in all, our experiences in working with the foundations, is much along the lines of what one could expect based upon studies of success factors for software measurement.

6.7 External Evaluation

After the end of the action research cycles, an external consultant evaluated the research project together with the participating companies (Tallqvist and Saranen, 2000). The purpose of this evaluation was to assess the potential for commercialization of the toolset. Since the results are interesting also from the point of evaluating the end-result of the research and provide an "unbiased" evaluation of the outcome of the research discussed in this thesis, I have decided to include the partial results of that study that are interesting from the point of view of implementing control panels, and leave, e.g., the analysis of markets and pricing out. To this end, I next discuss the results of the external evaluation with respect to the overall impact of the cooperation, the quality of the toolset, and the estimated maintenance effort needed if deploying the tool. Methodologically we did not have any control over the work of the external consultant. The results discussed below were obtained by interviewing the key stakeholders in the companies, which typically were the same people that we had worked closely with during three years.

[6] As discussed in the case description: the efforts at defining corporate goals for derivation of measurement led to an "explosion" of the number of goals, that we together with the company were not able to prioritize.

6.7.1 Post-Project Control Panel Usage

As previously discussed, only SoftCorp, deployed the toolset, and continued to use it after the end of the joint project. The three other companies did not use the tools, and reported various reasons for not doing so. TeleCorp considered the toolset technically unfinished, and expressed concern regarding future support and development. ElectroCorp reported that they did not take the tool into use due to a lack of resources and the lack of future support for the toolset. SecuriCorp, finally, reported that they did not have the resources to deploy the tool. The concern regarding future support for the toolset was mentioned by all companies, and is of course legitimate, but is outside the scope of this study, since we focussed on validating the idea of control panel definition and implementation, and did not aim for wide-scale deployment.

The concern regarding lack of resources might be understood as reflecting a multitude of concerns. First, the development and implementation of control panels turned out to be much more resource demanding than expected by both researchers and corporate partners. Furthermore, in most companies we were not able to develop control panels that were felt to be useful and worthwhile to take into use. Thus, the comment also reflects an underlying assumption that the cost-benefit ratio of implementing control-panels is not good enough to merit the investments needed.

When asked about the obstacles to deployment, respondents identified four factors: *corporate bureaucracy*, *management commitment*, *lack of ideological background information and marketing*, and *organizational change*.

Corporate bureaucracy, a factor particularly salient in the TeleCorp case can be a difficult obstacle to overcome when trying to implement control panels. In TeleCorp, we had difficulties getting access to corporate level databases, due to this. Also, due to the change of management and level of bureaucracy, the organization was not ready to deploy control panels. In the smaller organizations, bureaucracy was less of a problem.

Management commitment was problematic in the ElectroCorp, TeleCorp and SecuriCorp cases. In ElectroCorp, the CEO was committed to the project, but the person responsible for it was not. In TeleCorp, we had a similar situation—the person who wanted the control panel idea to work, and who also recently had headed the organization's balanced scorecard effort was promoted to head of a different division. His successor never showed any interest in the project. In the SecuriCorp case, the development manager was committed, but he did not manage to secure real support from the CEO or the marketing organization. The lack of management commitment was reported to lead to problems like lack of strategic direction and resources.

Lack of ideological background information and marketing, is a factor closely related both management commitment and organizational change, and in our own analysis, I did not present it separately. By this, the respondents meant that the idea of control panels and their implementation had not been sufficiently disseminated in the organizations. In all organizations, we did, however, have

both workshops and talks to personnel about the project. Thus, as such, this comment was interesting.

Organizational change, finally, affected the work in all pilots. SoftCorp was sold, a fact that later ended the deployment of control panels. ElectroCorp underwent two organizational reorganizations during our pilot, and SecuriCorp decided to downsize its development team. In TeleCorp, the divisional manager who originally decided to participate in the project moved to head another subsidiary. In all cases, this led to a need for reselling and re-establishing management commitment. As can be seen in the above comments, this was only partly successful. Also, organizational change draws a lot of the energy in an organization, which can make improvement projects like the one described here relatively less important than other ones. Also, since measurement ties directly into organizational processes and organizational structure, instituting a measurement system for an organization might be more appropriate during a period of relative stability. However, one could also take the point of view that a working measurement infrastructure, which can be used to measure whatever organizational structures, processes and products currently are operational could be a valuable asset even during periods of organizational instability. However, our experiences do not support this view.

These results, fit closely with our own analysis, discussed in the previous section. We will next turn to the impact of the cooperation, as evaluated by the corporate partners.

6.7.2 Impact

The pilot companies were asked to evaluate the impact of our collaboration on a five-point Likert scale with 1 representing "no impact" and 5 representing "major impact". The results are shown in Table 6.11.

Table 6.11: Impact of the Lucos project

Characteristic	Evaluation (mean)
Availability of internal information	4,0
Goal-orientation	3,8
Efficiency	3,0
Image	3,0
Bureaucracy	2,5
Customer satisfaction	2,4
Cost awareness	2,3
Quality awareness	2,0
Sales and profit	1,7

Respondents considered, unsurprisingly, that the areas in which there was the most potential for impact was the quality of company-internal information and goal-orientedness. As expected, the companies also felt that there was some impact on efficiency, as well as on corporate image. Having a measurement system that works well and is effectively and efficiently communicated to all stakeholders

should indeed improve efficiency. The impact on corporate image is explained by the fact that the companies early on indicated that they would clearly communicate their modern way of managing development to both partners and customers.

Of the evaluations on the other dimensions listed in Table 6.11, two were surpirising and worth further comments: cost awareness and quality awareness. I was surprised that they were evaluated so low: one would think that having cost and quality information available on-line at any time would help increase the awareness. Unfortunately, I do not have explanatory data. One could hypothesize that it is due to the fact that cost and quality awareness already are at such high a level in the companies that they cannot, or need not, be further raised. Alternatively, measurement and visualization were not considered important mechanisms for this purpose.

6.7.3 Toolset

During the evaluation by the external consultant, the pilot companies were asked to evaluate several properties of the visualization tool on a five-point Likert scale, with 1 representing an evaluation of "poor", 3 representing "neutral", and 5 "excellent". The results are shown in Table 6.12.

Table 6.12: Pilot companies' evaluation of the visualization tool

Characteristic	Evaluation (mean)
Development potential	4,4
Versatility	3,9
Progressivity	3,8
Tailorability	3,8
Technical quality	3,2
Complexity	3,0
Ease of use	3,0

As can be seen in the table, the pilot companies felt that the tool had a good potential for being evolved into a genuinely useful tool. The most critical success factor was considered to be the understanding of and integration with existing corporate data storage. The tool was also considered to be both versatile, progressive and tailorable. Thus, these objectives seem to have been well achieved. The technical quality was rated slightly positive, with one company giving a negative rating due to database interface problems.

Complexity received a neutral mean. This is explained by the fact that companies felt that the basic tool was easy enough to understand, but that the implementation of new measures and new visualizations was considered overly difficult. The same explanation also goes for usability: basic usage of the tool, i.e., looking at predefined panels and navigating between them, was considered easy. The process of defining and implementing measures was considered hard. Thus, the usability goal can be considered partly achieved.

As previously shown in Table 3.3, the tool was taken into real usage in only one company. In that company, the usage was reported to have a large impact, both on the pilot software development project in which it was initially used, as well as in the customer service function, where it was later deployed. In the other cases, the impact of the work was neglible, as the tool and ideas never made it into real use. However, the single successful case can be considered to show the *feasibility* of defining and implementing control panels, at least in small organizations, in highly visible projects.

Table 6.13 shows the pilots' evaluation of the need for maintenance and support related activities in case of toolset rollout. The evaluations were made on a five-point Likert scale with 1 standing for "high" and 5 standing for "low". Thus, the lower the mean, the better. The table is therefore sorted according to ascending mean.

Table 6.13: Pilot companies' evaluation of maintenance effort

Characteristic	Evaluation (mean)
Documentation effort	2,2
Administration effort	2,6
Day-to-day support effort	2,8
Training effort	3,2
Maintenance costs	3,8

The highest costs were considered to be those related to tool administration and documentation. Several companies felt that usage of the tool would require one full-day administrator. In addition, the detailed company-specific need for documentation was felt to be important. This was no surprise — implementing any tool in an organizational context requires adequate support, and any measurement program needs to be well documented. Also the need for day-to-day support was considered crucial, and therefore to incur non-neglible cost. However, this finding is not completely supported by our experiences; in the SoftCorp case, the actual effort for administering the tool after the measures and control panels had been developed and tested was very small.

Of less concern was the cost needed for training and technical maintenance of the tool. The pilot companies considered external technical maintenance to be crucial, since they did not want to start maintaining and developing the tool themselves. Since using the visualization tool to view existing panels or to develop own panels from existing visualizations was considered easy, the costs for training were estimated to be small.

6.8 Conclusions

This chapter discussed our experiences in defining and implementing control panels in four Finnish companies. The toolset was successfully piloted in all companies, however only one company used the toolset in production use, and this only

for a short period of three weeks. The cross-case analysis discussed these, somewhat disappointing results, and tentatively identified success factors and obstacles to control panel implementation. Most saliently, problems were related to the building of a working measurement foundation upon which control panels can be built. Constructing such a foundation requires substantial commitment from the organization, but is possible even in small companies, as the SoftCorp case showed. Organizational inertia, and organizational change also pose challenges to control panel implementation. We also identified success factors, the most critical of which was sustained management commitment.

Chapter 7

Conclusions

> The more research I do the more I find everything is at random. Somebody goes off in this direction, somebody in that, and who knows what the end result is going to be?
>
> Harrison E. Salisbury

7.1 Introduction

This chapter summarizes the work presented, identifies the contributions made in the thesis, evaluates the work, as well as presents topics for further research.

7.2 Summary

This dissertation has presented concepts, guidelines and a toolset for developing visual displays of software development status, "control panels", as well as discussed our empirical experiences in applying them in four Finnish high-technology companies.

The conceptual framework identifies a minimal set of concepts needed to build hierarchies of navigable control panels and via the *measure* concept, provides a linkage to corporate measurement programs without tying into any particular measurement development approach. The framework is evaluated below.

In addition to the framework, the work identified basic activities that needs to be performed *in addition to* normal measurement program development activities, when defining and implementing software development control panels.

The toolset discussed provides support both for building visualizations and control panels and for gathering data upon their use. This data can be used to better understand usage patterns for measurement systems.

Finally, the empirical experiences show the initial feasibility of implementing software development control panels, but also show that it is a major undertaking that require a substantial degree of commitment as well as adequate resourcing.

Successful control panel implementation is shown to be able to provide value to an organization, though we are unable to separate the value of visualizations from measurement program implementation.

7.3 Contributions

This thesis makes three distinct contributions to knowledge. First, the vision and framework presented in Chapter 4 presents a way of conceptualizing and structuring the presentation of measurement results in software development organizations. While the focus in this research has been on software development, the concepts and development process is likely to be applicable also to other fields, like e.g. new product development.

Second, the toolset discussed in Chapter 5 provides a platform that can be used both for developing and supporting control panels for software development, as well as for analysing their use, thus deepening our understanding of the benefits and drawbacks.

Finally, our empirical experiences, discussed in Chapter 6, show that implementing control panels is viable, but challenging. In particular, tool deployment and integration into existing measurement programs and tools is challenging and demands a high level of organizational commitment and easily a substantial investment. We identified tentative success factors and issues that must be dealt with when trying to implement control panels.

7.4 Evaluation of the Research

In this section, I evaluate the research discussed in the thesis. When evaluating research, one can use general criteria for science, such as *objectivity, criticality, autonomy* and *progressivity* (Niiniluoto, 1984). While these general criteria can be useful, particular research approaches and scientific fields typically use more practical guidelines. Since the research in this thesis is mainly constructive, but there is little guidance for research evaluation in the field of software engineering, I have opted to use a combination of the criteria presented by (Kasanen et al., 1993) and the guidelines for design research presented by Hevner et al. (2004) and discussed in Section 3.3. The discussion is structured according to the design research guidelines, with aspects added from other sources as needed.

7.4.1 Design as an Artifact

The first design research guideline states that "Design science is supposed to produce a viable artifact in the form of a construct, a model, a method, or an instantiation". Since the overall goal of design research is to product an artifact, the evaluation of the actual artifacts is a logical place to start the evaluation.

In the research presented in this thesis, several artifacts were described. First, an overall idea of *software development control panels* was described, and the *concepts* needed to develop these, as well as a *framework* that can be helpful when identifying

what kind of panels to build was described. A tentative *process* for developing and evolving a corporate set of control panels was also described. Finally, a *measurement toolset* was described. I next evaluate each of these as research artifacts, first discussing, as is typically done when evaluating constructive research, how well the requirements regarding their development was met, and then their viability.

7.4.1.1 Conceptual Framework and Guidelines

We had seven requirements for the *conceptual framework and guidelines* as discussed in Chapter 4, and relisted in Table 7.1 together with an evaluation of how well we achieved them.

When working in the case organizations, we found it to be harder than we had expected to systematically address the development of control panels. As described in the previous chapter, using the action research cycle, we ended up working in ways that often were quite specific to the individual organizations. The theoretical constructs that received the most "testing"—though they partly can be seen as an outcome since I did not make them explicit until the end of the project—were the basic concepts and the *logical view of a software development project organization.* In particular, we found the concepts simple and natural to use, and the logical organizational view to be useful both when trying to understand the "big picture" of what goes on in a software development organization, and for structuring the development of control panels.

7.4.1.2 Toolset

We had ten initial requirement for tool development. These, as well as how they have been met is summarized in Table 7.2. As shown in the table, we can consider tool development a success with respect to the requirements, since we were able to meet all of them. Our empirical experience with the toolset showed its viability in the sense that we were able to develop and deploy control panels in one of the four case organizations, and develop example visualizations using corporate data in all of them.

Finally, the toolset also helped show the viability of the vision since it can be considered an instantiation of it, and thus works as "proof by construction" (Nunamaker et al., 1991).

7.4.2 Problem Relevance

In addition to producing a viable artifact, design research should work on important problems. Kasanen et al. (1993) suggests that the relevancy of the problem should be evaluated both from a practical and a theoretical point of view.

In Chapter 2 I discussed the *theoretical relevance* of the problem and showed that while there is a lot of literature on software measurement in general, and on measure and measurement program development in particular, issues of measurement presentation, presentation grouping, and tool support have received quite little research attention. Thus, there is a clear theoretical space for frameworks,

Table 7.1: Evaluation of developed framework vs. requirements

No	Requirement	Achievement
FR-1	*Concepts:* The framework should define the central concepts needed for the development of control panels.	Achieved. The conceptual framework discussed in Section 4.3 does this.
FR-2	*Guidance:* Guidelines for control panel development should be developed.	Partly achieved. The list of possible partitionings, the logical view of a software development organization as well as the tentative development process (sections 4.4 and 4.5 respectively) address this.
FR-3	*Method independence:* Control panel development should be independent of measurement paradigm or methodology.	Partly achieved. The conceptual framework for control panels does not tie into any particular measurement paradigm. However, we strongly believe in *stakeholder* and *goal-oriented measurement* as evident from the discussion.
FR-4	*Organizational independence:* It should be possible to develop control panels for different organizations and environments.	Partly achieved/Unverified. We implemented control panels in one organization only, but piloted the idea in the others as well.
FR-5	*Technological independence:* Control panel development should be independent of the particular technological tools deployed.	Achieved. It is possible to use, e.g., a spreadsheet tool to develop control panels.
FR-6	*Usability:* The framework should be easy to learn and use.	Unverified. The final versions of the constructs have not been tested in practice. However, they reflect the lessons learned during our work with the case organizations.
FR-7	*Feasibility:* Control panel development should be feasible in practice.	Partly achieved. We succeeded in developing very simple control panels in one organization.

Table 7.2: Evaluation of developed toolset vs. requirements

No	Requirement	Achievement
TR-1	*Stakeholder customizability:* The system should support the definition and display of different, customizable views for different stakeholders.	Achieved. Stakeholders can both build own charts, as well as build and configure own control panels.
TR-2	*Navigation:* The system should support navigation between different panels.	Achieved. Both static navigation (from charts, using buttons), and dynamic navigation (from datasets to panels) supported.
TR-3	*Access control:* The system should support authentication and access control.	Achieved. The system supports both.
TR-4	*Data source compatibility:* The system should be compatible with existing corporate measurement systems and data sources.	Achieved. The system stores replicated measurement data in its own database.
TR-5	*Multi-method approach:* The system should support different approaches to the definition and application of measurement.	Achieved. The system does not support or require any particular measurement paradigm or method.
TR-6	*Ease of use:* Basic system usage should be easy and not require special skills or training.	Achieved. Users reported that basic usage is easy.
TR-7	*Multi-platform compatibility:* The system should work in both Windows and Unix environments, and must support both the Netscape and Internet Explorer browsers.	Achieved. The system was developed in a mixed environment, and piloted on both Unix (Linux) and Windows machines.
TR-8	*Distributedness:* The system should support people working in physically separate locations.	Achieved. The Usage of WWW protocols and Java technology makes location into a non-issue. Bandwidth and firewalls can worsen the user experience.
TR-9	*Low-cost:* The system must be cheap to acquire and should not contain any costly third-party components.	Achieved. The system relied only on a cheap relational database engine that with some work can be replaced with databases from other vendors, and on a visualization library that did not have any distribution fees.
TR-10	*Logging:* The system should support logging of user actions.	Achieved. The system supports both user and system event logging.

tools and experiences in this area, establishing the theoretical relevance of the work discussed in this thesis.

The *practical relevance* of the problem solved was shown by using both literature and our experiences in contacting companies and discussing the idea of control panels with them. The practical relevance was further shown during the course of the research, and even after the project, when we have had several contacts from several interested companies.

7.4.3 Design Evaluation

The third guideline states that the utility, quality and efficacy of the design artifact should be rigorously demonstrated via well-executed evaluation methods. Since rigor, i.e., the way research is conducted, is discussed under its own guideline, here we are interested in the use of appropriate evaluation methods.

First, it is worth noting that, following the guideline, the research discussed here has been evaluated in industry. In fact, as discussed in Chapter 3, the constructs were simultaneously developed and evaluated in close cooperation with the industrial partners. Next, we need to discuss the appropriateness of the evaluation methods employed. When developing a new construct, the process is often (and should be) iterative (Kasanen et al., 1993; Hevner et al., 2004). Working closely with companies is a way of ensuring practical relevance of the problem and solution, and it helps weed out unfeasible solutions (Hevner et al., 2004). Thus, I think the case studies described can be considered appropriate for preliminary evaluation of the presented constructs.

Another issue worth discussing is whether the idea of combining design and action research in the way presented was a good strategy. Most case studies in software engineering seem to focus only on arguing in favor of the proposed construct, with little attention paid to the empirical set-up, the client-infrastructure, the role of the researcher, the research process and methodology or to problems and dead-ends met. While it can be argued that it is beneficial to the reader to focus only on the interesting thing—the construct and how novel it is and how well it works—it can rarely be said to fulfil general criteria for scientific reporting, in which, e.g., the possibility for other researchers to replicate the experience based upon the report is stressed. I think that framing the research using the process described in Chapter 3 was beneficial in that it makes explicit the context and process used in the case studies. This makes it easier for the reader to appreciate the results and experiences gained during the research. However, a problem with this kind of structure is the lengthy reporting it generates, in particular when doing multiple case studies, each containing several action research cycles. I tried to alleviate the problem by using summary tables, but this invariably leads to less enticing case study descriptions.

In our case studies, the researchers were intimately involved in developing, implementing and deploying the constructs. Thus, the researchers have a vested interest in showing that the constructs work (in the fear of otherwise appearing stupid). This *researcher bias* is made worse by the fact that most companies also

had people assigned to the joint project, making positive results important for the companies as well. Thus, in this research, as other research that presents preliminary validation case studies performed by the researcher who has developed a new construct, there is a clear risk of overstating the contributions and results.

We have tried to alleviate this problem by involving several people in the organizations in the evaluations, and by having external consultants do a follow-up study, as discussed previously. Despite these measures, there is a clear need for further validation of the ideas, concepts and toolset discussed in this thesis before any strong conclusions can be made.

7.4.4 Research Contributions

The research should provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies. As discussed above, this research presented several artifacts that can be considered contributions. These were discussed above in Section 7.3, and are not reiterated here.

7.4.5 Research Rigor

As all science, design science relies upon the application of rigorous methods in both the construction and evaluation of the design artifact. Rigor is achieved by using appropriate research methods in rigorous ways. The literature contains several suggestions for how to achieve this.

Patton (1990), for example, presents techniques for enhancing the quality of case studies, including the testing of rival explanations, the search for negative cases, triangulation (mixing methods, usage of several sources, multiple analysts, theory triangulation). In his often cited book on case study research, Yin (1994, pp.32–38) argues for the use of four criteria for evaluating the quality of case study research designs: construct validity, internal validity, external validity, and reliability. Yin also presents criteria for improving the quality of the case study: to have informants as well as peers review the study. Also, when evaluating a case study, he suggests the use of the following criteria: significance, completeness (which is hard to describe operationally), the consideration of alternative perspectives, the display of sufficient evidence, composed in an engaging manner. For improving construct validity, he suggests the use of multiple sources of evidence, to establish a chain of evidence, and to have key informants review draft case study reports. All of these take place in the data collection phase. Internal validity, an issue related to data-analysis, can be improved by doing pattern matching, explanation building, and time-series analysis. External validity, which is related to the research design, can be improved by the use of replication logic in multiple-case studies. Reliability, finally, can be improved by the use of a case study protocol, as well as by developing a case study database, both in the data collection phase.

In action research, the role of the researcher and his relationship to the client is crucial. In at least two cases, we failed to build the long-term trusting relationships with mutual respect that are necessary to successfully carry out this kind of research. As such, this is not a weakness of the framework, but the research

process, and the inexperience of the researchers both regarding the subject matter and regarding the process of doing research itself.

Methodologically, the research quite well conforms to the principles and process for constructive research presented in Chapter 3, in which also the data collection methodologies were discussed. We used several ways of improving the quality of the empirical work, including:

- *Triangulation.* We used both source and method triangulation (Jick, 1979). Source triangulation meant that we used several informants and company documentation. In particular in the initial analysis phase (diagnosis in the first action research cycle), we mixed qualitative (i.e. interviews and document analysis) and quantitative (document and database analysis) methods. In the SoftCorp case, in which we succeeded in deploying the tool, we also combined quantitative data (analysis of the MESS logs) with qualitative interviews.

- *Informant review.* Our informants were shown the reports we had written based upon their interviews and asked to correct them. This technique can be used to increase construct validity (Yin, 1994).

- *Replication logic.* While each case had distinguishing characteristics, they all aimed at the same goal, and thus can be considered—at a high level—as utilizing a replication logic that can help increase the construct validity of the work.

However, we also made some mistakes that proved expensive, especially from the point of writing this thesis. In particular, we did not keep research diaries or structured case notes, and we did not debrief ourselves after e.g. workshops or company meetings. Neither did we tape record the interviews we did with company personnel. All of these activities, while demanding little additional effort at the time of data collection, would have provided invaluable data from the point of view of analysing the case studies. Fortunately, we wrote reports and kept minutes at the meetings in the companies, which I used when analysing the cases.

Most of these weaknesses could have been overcome by basic training in research methodology. In the software engineering field, this has, at least up till now, not been considered important. Fortunately, this is changing, and more and more software engineering programs provide training in performing empirical research.

7.4.6 Design as a Search Process

The sixth, guideline deals with the fact that design typically is a search process, in which one aims at bridging the gap between a current state and a goal state by designing a construct. During this process, several possible design alternatives, or alternative solutions might exist, and it might be hard to evaluated which of them is optimal. It is, however important to use explicit criteria for assessing the goodness of a solution, and to look for an optimal solution if one exists.

In the research discussed here, there is likely to be a vast number of solutions for fulfilling the requirements for both the framework and the toolset construction. I do not claim optimality, but note that the developed solutions represent possible solutions to the stated problems. According to Kasanen et al. (1993), a major criteria for evaluating the goodness of constructive research, is the *practical utility* of the constructs, i.e., how well (effectively and efficiently) they work in real life. As stated earlier, I cannot claim more regarding practical utility than that the proposed ideas were feasibly implemented and taken into use in one single case company. Thus, they are not totally unfeasible.

7.4.7 Communication of Research

While perhaps not directly applicable to a thesis written as a monograph, the final guideline states that the research should be reported to both managerial and technical audiences. Various aspects of the research reported in this thesis has been published in conference papers, both from a technical point of view (e.g. Vanhanen et al. (1999)) and from a managerial point of view (e.g. Lassenius et al. (1999); Lassenius and Rautiainen (1999)). Thus, I consider that the demands of this guideline have been met sufficiently for a doctoral thesis. In addition to the publications, the research was disseminated through seminars for industrial audiences.

7.4.8 Evaluation Summary

To summarize the above discussion, the research presented in this work quite well fulfills the criteria suggested for design research: it has presented a novel solution to a problem that is both theoretically and practically relevant. The solution has been empirically evaluated as part of the design process, and the artifacts and evaluation reported. The main limitations of the research are related to research rigor, and the lack of empirical validation of the final constructs. Also, the lack of deployment experiences limit the ability to draw conclusions regarding the usefulness of the concepts and toolset. As always, a case study approach provides limited opportunities for generalization. In addition, in this kind of "constructive action research", the role of the researcher and his or her interactions with the client play and important role that in addition to the actual constructs can influence the results paramountly. Despite these weaknesses, I think that the concepts and framework as such can at least provide some guidance to other researchers and companies who might be interested in developing control panels for their development efforts. The toolset, as an instantiation of the constructs, provides a "proof-by-construction" of the ideas and therefore also can be considered a contribution.

7.5 Future Work

The research presented in this thesis can be continued in several ways. Existing visualizations in the field of software engineering could be systematically classified

and analyzed. An empirical investigation could assess the subjective value of different visualizations to various stakeholders. The information from such a study could be used to develop guidelines for what kinds of visualizations different stakeholders are interested in.

Gaining more empirical experience with the use of on-line visual feedback would be valuable. The tool presented can be used to empirically study the use of panels in real life.

Despite the importance of visual presentations for communicating measurement results, the field contains almost no guidance on how to develop good presentations. Such guidelines accompanied with examples, could provide valuable help to practitioners implementing measurement programs. Such knowledge could also be used to develop visualization libraries.

The ideas and tools presented here could be applied to other processes and situations. The ideas are quite general, and very little confine their usefulness to software engineering. Thus, they could be applied to the development and implementation of performance measurement systems in general.

Distributed and outsourced software development is becoming increasingly common. In such projects, monitoring progress and getting up-to-date information is even harder than in traditional software development. Augmenting the work to accomodate definition and implementation of cross-organizational measurement systems might provide worthwhile. The tool architecture as such already supports distribution.

Additional clients could be developed, e.g., for mobile phones or PDAs. Modern wireless technology could be utilised to make development status available "anywhere" and "anytime".

Finally, the discussion on the activities for developing control panels defined in Chapter 5 could be expanded and concrete guidelines for performing them developed. Further empirical experiences could provide interesting illustrations to such guidelines.

Bibliography

A. Abran, J. W. Moore, P. Borque, and R. Dupuis, editors. *Guide to the Software Engineering Body of Knowledge: 2004 Edition — SWEBOK.* IEEE Computer Society, 2004.

Anonymous. *Sybase Enterprise Data Studio Feature Guide.* Sybase Inc., 1998a.

Anonymous. *A Guide to the Project Management Body of Knowledge (PMBOK) Guide.* Project Management Institute, Newtown Square,PA, 2000a.

Anonymous. The program manager's guide to software acquisition best practices. Handbook Version 2.31, Computers & Concepts Associates, 1998b.

Anonymous. The software project manager's control panel, v. 2.31. `http://www.spmn.com/`, 2000b.

D. Avison, F. Lau, M. Myers, and P. A. Nielsen. Action research. *Communication of the ACM*, 42(1):94–97, January 1999.

V. Basili. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, 1984.

V. Basili. The role of experimentation in software engineering: Past, current, and future. In *Proceedings of ICSE-18*, page 442–449. IEEE Computer Society Press, Los Alamitos, CA, 1996.

V. Basili and H. Rombach. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 11(6):758–773, 1988.

V. Basili, R. W. Selby, and D. H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, SE-12(7):733–743, 1986.

V. Basili, G. Caldiera, and H. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering.* Wiley, 1994.

V. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, 1999.

R. L. Baskerville and A. T. Wood-Harper. A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11:235–246, 1996.

T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext transfer protocol - http/1.0. Technical report, MIT/LCS, UC Irvine, 1996/05// 1996.

M. Berry and R. Jeffery. An instrument for assessing software measurement programs. *Empirical Software Engineering*, 5(3):183–200, 2000.

L. Briand, C. M. Differding, and H. Rombach. Practical guidelines for measurement-based process improvement. *Software Process - Improvement and Practice*, 2:253–280, 1996.

J. Brooks, F.P. *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley, Reading, MA, USA, 20th anniversary edition with four new chapters edition, 1995.

N. Brown. Industrial-strength management strategies. *IEEE Software*, page 94–103, 1996.

A. Burr and M. Owen. *Statistical Methods for Software Quality: Using Metrics to Control Process and Product Quality.* International Thomson Computer Press, 1996.

D. Campbell and J. Stanley. *Experimental and Quasi-Experimental Designs for Research.* Rand McNally College Publishing Company, Chicago, 1963.

D. N. Card. What makes for effective measurement. *IEEE Software*, 10(6):94–95, November 1993.

V. Chiesa, P. Coughlan, and C. Voss. Development of a technical innovation audit. *Journal of Product Innovation Management*, 13(2):105–136, 1996.

R. Cooper. *Winning at New Products.* Addison-Wesley, Reading, MA, USA, 2nd edition, 1993.

R. Cooper, S. Edgett, and E. Kleinschmidt. *Portfolio Management for New Products.* Addison-Wesley, Reading, Massachusetts, 1998.

I. Crnkovic and P. Willfor. Change measurements in an scm process. In *Proceedings of SCM–8, the 8th International System Configuration Management Workshop*, 1998.

B. Curtis. Measurement and experimentation in software engineering. *Proceedings of the IEEE*, 68(9):1144–1157, 1980.

R. Curtis and S. Scarfone. Xface, an x tool for presenting multivariate data, and its use with software metrics. In *Eleventh Annual International Phoenix Conference on Computers and Communications*, page 525–530, 1992.

M. A. Cusumano and R. W. Selby. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets and Manages People.* The Free Press, 1995.

M.-K. Daskalantonakis. A practical view of software measurement and implementation experiences within motorola. *IEEE Transactions on Software Engineering*, 18(11):998–1010, 1992.

C. Debou, A. Kuntzmann-Combelles, and A. Rowe. A quantitative approach to software process management. In *Proceedings 2nd International Software Metrics Symposium*, page 26–34. IEEE Computer Society Press, 1994.

I. Distributive. Datadrill dashboard factsheet, 2005. URL `http://www.distributive.com/software_dashboard_factsheet.html`.

R. Dolan and J. Matthews. Maximizing the utility of customer product testing: Beta test design and management. *Journal of Product Innovation Management*, 10 (4):318–330, 1993.

R.-R. Dumke. Came tools-lessons learned. In *Proceedings of the Fourth International Symposium on Assessment of Software Tools*, page 113–114, 1996.

R.-R. Dumke and H. Grigoleit. Efficiency of came tools in software quality assurance. *Software Quality Journal*, 6(2):157–169, 1997.

R.-R. Dumke and A.-S. Winkler. Came tools for an efficient software maintenance. In *Proceedings of the First Euromicro Conference on Software Maintenance and Reengineering, 1997,"(EUROMICRO 97)"*, page 74–81, 1997.

R. Englund and R. Graham. From experience: Linking projects to strategy. *Journal of Product Innovation Management*, 16(1):52–64, 1999.

N. Fenton and M. Neil. Software metrics: A roadmap. In *Proceedings Conference on the Future of Software Engineering*, ICSE-2000, page 357–370. ACM Press, 2000.

N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, London, 2nd edition, 1997.

N. Fenton, S. L. Pfleeger, and R. L. Glass. Science and substance: A challenge to software engineers. *IEEE Software*, 11(4):86–95, July 1994.

W. A. Florac and A. D. Carleton. *Measuring the Software Process: Statistical Process Control for Software Process Improvement*. Addison-Wesley, Reading, MA, 1999.

W. A. Florac, R. Park, and A. Cerleton. Practical software measurement: Measuring for process management and improvement. Technical report, Software Engineering Institute, 1997/04// 1997.

E. Foltin and R.-R. Dumke. Aspects of software metrics database design. *Software Process Improvement and Practice*, 4(1):33–42, 1998.

A. Fuggetta, L. Lavazza, S. Morasca, S. Cinti, G. Oldano, and E. Orazi. Applying gqm in an industrial software factory. *ACM Transactions on Software Engineering and Methodology*, 7(4 L1 - c:/Library/Fuggetta98.pdf):411–448, 1998.

T. Gilb. *Software Metrics*. Winthrop Publishers, Cambridge, MA, 1977.

R. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44:491–506, 2002.

R. L. Glass. A comparative analysis of the topic areas of computer science, software engineering, and information systems. *Journal of Systems and Software*, 19 (3):277–289, 1992.

R. L. Glass. The software research crisis. *IEEE Software*, 11(6):42–47, November 1994.

W. Goethert and M. Fisher. Deriving enterprise-based measures using the balanced scorecard and goal-driven measurement techniques. Technical Report CMU/SEI-2003-TN-024, CMU/SEI, 2003.

W. Goethert and W. Hayes. Experiences in implementing measurement programs. Technical Report CMU/SEI-2001-TN-026, Software Engineering Institute, 2001/// 2001.

W. Goethert and J. Siviy. Applications of the indicator template for measurement and analysis. Technical Note CMU/SEI-2004-TN-024, CMU/SEI, 2004.

D.-R. Goldenson, A. Gopal, and T. Mukhopadhyay. Determinants of success in software measurement programs: initial results. In *Proceedings Sixth International Software Metrics Symposium*, page 10–21, 1999.

R. B. Grady. *Practical Software Metrics for Project Management and Process Improvement.* Prentice-Hall, Englewood Cliffs, NJ, 1992.

R. B. Grady and D. L. Caswell. *Software Metrics: Establishing a Company-Wide Program.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

E. Gummesson. *Qualitative methods in management research.* Sage, Thousand Oaks, CA, 2000.

A. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.

W. Humphrey. *A Discipline for Software Engineering.* Addison-Wesley, Reading, MA, USA, 1995.

W. Humphrey. The personal software process (psp). Technical report, Software Engineering Institute, 2000/// 2000a.

W. Humphrey. The team software process (tsp). Technical report, Software Engineering Institute, 2000/// 2000b.

W. Humphrey. *Managing the Software Process.* Addison-Wesley, Reading, MA, USA, 1989.

T. Jick. Mixing qualitative and quantitative methods: Triangulation in action. *Administrative Science Quarterly*, 24(4):602–611, 1979.

P. M. Johnson, H. Kou, M. Paulding, Q. Zhang, A. Kagawa, and T. Yamashita. Improving software development management through software project telemetry. *IEEE Software*, 22(4):76–85, July-August 2005.

N. Juristo and A. M. Moreno. *Basics of Software Engineering Experimentation.* Kluwer Academic Publishers, Boston, MA, 2001.

R. Kaplan and D. Norton. The balanced scorecard - measures that drive performance. *Harvard Business Review*, 70(1):71–79, 1992.

R. Kaplan and D. Norton. Putting the balanced scorecard to work. *Harvard Business Review*, 71(5):134–147, 1993.

E. Kasanen, K. Luukka, and A. Siitonen. Konstruktiivinen tutkimusote liiketaloustieteessä. *Liiketaloudellinen Aikakauskirja*, 40(3):301–327, 1991.

E. Kasanen, K. Lukka, and A. Siitonen. The constructive approach in management accounting research. *Journal of Management Accounting Research*, 5(Fall):243–264, 1993.

R. Kempkens, P. Rosch, L. Scott, and J. Zettel. Instrumenting measurement programs with tools. In *Product Focused Software Process Improvement*, 2000.

I. C. Kerssens-van Drongelen. *Systematic Design of R&D Performance Measurement Systems.* Phd, University of Twente, 1999.

B. Kitchenham. *Software Metrics: Measurement for Software Process Improvement.* Blackwell, Oxford, UK, 1996.

B. Kitchenham. Procedures for undertaking systematic reviews. Unpublished technical report, May 2004.

B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. E. Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, 2002.

C. Lassenius and K. Rautiainen. An incremental approach for improving the controllability of product development. In *Proceedings 6th International Product Development Management Conference.* EIASM, 1999.

C. Lassenius, M. Nissinen, K. Rautiainen, and R. Sulonen. The interactive goal panel: A methodology for aligning r&d activities with corporate strategy. In *Proceedings of the 1998 IEEE Conference on Engineering Management.* EIASM, 1999.

L. Lavazza. Providing automated support for the gqm measurement process. Technical report, CEFRIEL, 1998/// 1998.

L. Lavazza. Providing automated support for the gqm measurement process. *IEEE Software*, 17(3):56–62, 2000.

McCall. Factors in software quality. Technical report TR-77-369, Rome Air Development Center, USAF, 1977.

C. Meyer. How the right measures help teams excel. *Harvard Business Review*, 72 (3):95–103, 1994.

G. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers.* HarperCollins Publishers, 1991.

J. Münch and J. Heidrich. Software project control centers: Concepts and approaches. *Journal of Systems and Software*, 70(1–2):3–19, 2004.

N. Nambisan and D. Wilemon. Software development and new product development: Potentials for cross-domain knowledge sharing. *IEEE Transactions on Engineering Management*, 47(2):211–220, 2000.

I. Niiniluoto. *Johdatus tieteenfilosofiaan.* Otava, Helsinki, 1984.

J. F. J. Nunamaker, M. Chen, and T. D. Purdin. Systems development in information systems research. *Journal of Management Information Systems*, 7(3):89–106, Winter 1990–91 1991.

R.-J. Offen and R. Jeffery. Establishing software measurement programs. *IEEE Software*, 14(2):45–53, 1997.

R. Park, W. Goethert, and W. A. Florac. Goal-driven software measurement - a guidebook. Technical report, Software Engineering Institute, 1996/// 1996.

P. Parviainen, J. Jarvinen, and T. Sandelin. Practical experiences of tool support in a gqm-based measurement programme. *Software Quality Journal*, 6(4):283–94, 1997.

M. Q. Patton. *Qualitative evaluation and research methods.* Sage Publications, Newbury Park, Calif., 2nd edition, 1990.

M. C. Paulk. *The Capability maturity model: guidelines for improving the software process.* Addison-Wesley, Reading, MA, 1995.

S. L. Pfleeger and C. McGowan. Software metrics in the process maturity framework. *Journal of Systems and Software*, 12(3):255–261, 1990.

C. Potts. Software engineering research revisited. *IEEE Software*, 10(5):19–28, 1993.

K. Pulford, A. Kuntzmann-Combelles, and S. Shirlaw. *A Quantitative Approach to Software Management: The ami Handbook.* Addison-Wesley, Wokingham, England, 1996.

L. Putnam and W. Myers. *Industrial Strength Software: Effective Management Using Measurement.* Los Alamitos, CA, USA, 1997.

K. Rautiainen, C. Lassenius, J. Vähäniitty, J. Vanhanen, and M. Pyhäjärvi. A tentative framework for managing software product development in small companies. In *Proceedings 35th Annual Hawaii International Conference on the System Sciences.* IEEE Computer Society Press, 2002.

K. Risku and T. Rinta-Aho. Proactive caching of database queries for web use. Technical report, Helsinki University of Technology, 1999/// 1999.

W. Royce. Managing the development of large software systems. In *Proceedings of the IEEE Westcon*, page 1–9. IEEE Computer Society Press, Los Alamitos, CA, 1970.

C. B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, 25(4):557–572, July/August 1999.

M. Shaw. Prospects for an engineering discipline of software. *IEEE Software*, 7(6): 15–24, 1990.

A. J. Shenhar. From theory to practice: Toward a typology of project-management styles. *IEEE Transactions on Engineering Management*, 45(1):33–48, 1998.

R. Simons. *Levers of Control: How Managers Use Innovative Control Systems to Drive Strategic Renewal.* Harvard Business School Press, Boston, MA, USA, 1995.

R. Simons. *Performance Measurement & Control Systems for Implementing Strategy.* Prentice Hall, Upper Saddle River, NJ, 2000.

P. G. Smith and D. G. Reinertsen. *Developing Products in Half the Time: New Rules, New Tools.* Van Nostrand Reinhold, New York, NY, USA, 1998.

v. R. Solingen and E. Berghout. *The Goal/Question/Metric Method.* McGraw-Hill International (UK), Maidenhead, 1999.

v. R. Solingen and E. Berghout. Integrating goal-oriented measurement in industrial software engineering: Industrial experiences with and additions to the goal/question/metric method (gqm). In *Proceedings Seventh International Software Metrics Symposium*, page 246–258. IEEE Computer Society Press, 2001.

K.-G. Sprague. The role of software configuration management in a measurement-based software engineering program. *SIGSOFT Software Engineering Notes*, 16 (2):62–6, 1991.

E. T. Stringer. *Action Research.* SAGE Publications, Thousand Oaks, CA, USA, 2nd edition, 1999.

G. I. Susman and R. D. Evered. An assessment of the scientific merits of action research. *Administrative Science Quarterly*, 23(4):582–603, December 1978.

K. Tallqvist and H. Saranen. Lucos: Kaupallistamismahdollisuuksien selvittely, loppuraportti. it.management, unpublished report (in Finnish), 2000.

J. Tian, J. Troster, and J. Palma. Tool support for software measurement, analysis and improvement. *Journal of Systems and Software*, 39(2):165–78, 1997.

W. F. Tichy. Should computer scientists experiment more? *Computer*, 31(5):32–40, 1998.

W. F. Tichy, P. Lukowicz, P. Lutz, and E. A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems Software*, 1995(28): 9–18, 1995.

E. Tufte. *Visual Explanations.* Graphics Press, 1997.

E. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 2nd edition, 2001.

J. Vanhanen, C. Lassenius, and K. Rautiainen. The visualisation client applet - a tool for increasing the visibility of product development. In *Proceedings of the 6th International Product Development Management Conference*, Cambridge, UK, 1999.

J. Vanhanen, C. Lassenius, and K. Rautiainen. Combining data from existing company data sources: Architecture and experiences. In *Knowledge Management Systems: Theory and Applications.* Kluwer, 2000.

E. F. Weller. Practical applications of statistical process control. *IEEE Software*, 17(3):48–55, 2000.

S. Wheelwright and K. Clark. *Revolutionizing Product Development.* The Free Press, New York, 1992.

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering.* Kluwer Academic Publishers, Boston, MA, 2000.

R. K. Yin. *Case Study Research: Design and Methods.* SAGE Publications, Thousand Oaks, CA, USA, 2nd edition, 1994.

M. Zelkowitz and D. R. Wallace. Experimental models for validating technology. *Computer*, 31(5):23–31, 1998.

H. Zuse. History of software measurement, September 1995. URL `http://irb.cs.tu-berlin.de/~zuse/metrics/History_00.html`.

ISBN 951-22-8196-1
ISBN 951-22-8197-X (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)