Miroslaw Staron
Wilhelm Meding

Software Development Measurement Programs

Development, Management and Evolution

Springer

Software Development Measurement Programs

Miroslaw Staron • Wilhelm Meding

Software Development Measurement Programs

Development, Management and Evolution

Springer

Miroslaw Staron
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden

Wilhelm Meding
Ericsson AB
Gothenburg, Sweden

ISBN 978-3-319-91835-8 ISBN 978-3-319-91836-5 (eBook)
https://doi.org/10.1007/978-3-319-91836-5

Library of Congress Control Number: 2018945114

To our families. . .

Foreword by Tom Gilb

Engineering Metrics Are a Necessary Tool for Software Development

The biggest and longest-lasting problem in software development is that we constantly fail to live up to our own promises and stakeholder expectations.

There is a simple explanation. We have in the past six decades seen an explosive growth in size, complexity, and the need for integration with other disciplines. At the same time, it takes a long time, decades to centuries, to develop successful cultures to handle such complexity, and we have not had enough time; by a long shot. In fact the nature of the problem seems to still be changing, faster than any time scale we would need to master, and cope with, the complexity and growth.

Some organizations, such as my own clients, Ericsson, IBM, HP, Intel and Boeing, have had to face size, complexity and change earlier than many others, and this is where this book has a role to play. This book incorporates many of the experiences and fruits of efforts to try to cope with the problems.

Not really surprisingly the answer to these problems, as in ages past (think Roman engineering, Egyptian pyramids), lies in the discipline of engineering. It is no accident that almost all of organizations such as the above-mentioned ones, have had a strong engineering culture, before the arrival of "software" as a component. And they have learned to apply engineering numeracy to the software domain.

The key tool in engineering, as in science, is *numeracy*. Quantification, measurement, feedback, learning, adjustment: a fact-driven culture. An 'evidence-based' culture. Too much of software culture today is based on the *craft of programming*, not the engineering of software (the *term* 'Software Engineering' was coined in 1968). The craft does not scale!

"Those who cannot remember the past are condemned to repeat it." (Santayana, 1905)

This book is *advanced, practical and useful* to potential readers, both industrial and academic.

Your own organization and problems are no doubt different from the organizations at the root of this book. But you can still select key ideas and practices, and adapt them to your current needs. You can be inspired to take new and advanced steps forward.

I certainly would recommend this book to people who want a deeper insight into the software metrics of today, and the trends toward tomorrow's software measurement.

With tools like these to build on, we might some time catch up with the complexity of the software development problem. It is not the whole answer, but we cannot afford to re-invent this wheel; we must learn from it.

Oslo, Norway Tom Gilb
www.Gilb.com
February 2018

Foreword by Alain Abran

In both public- and private-sector organizations, huge amounts of money are spent on software, competing for scarce organizational resources: how do you ensure that such amounts are spent wisely and efficiently?

In engineering and in business, as well as in all applied sciences including medicine and social sciences, measurement is a fundamental tool. However, unlike these traditional fields, which have access to mature measurement programs based on international standards, the software community at large is still looking for evidence of well-designed software measurement programs with proven value. In the software domain in particular, there is a large variety of software metrics proposed in academia and by tool vendors for monitoring software development and evolution; however, many of them have fairly weak foundations and many will not withstand the test of time. Implementing software measurement programs in an organization therefore requires addressing a number of issues, including: Which software metrics options to choose? In what contexts should they be used? And what are the conditions of success for the design and deployment of software measurement programs? Such are the issues discussed by the authors of this book.

Staron and Meding share their years of experience from collaboration between industry and academia in the design of software measurement programs and their successful deployment and use as decision-making tools. The authors have structured their measurement programs based on the wealth of accumulated knowledge in mature fields of measurement, including the International Vocabulary on Metrology and the ISO 15939 standards for software measurement programs. They also share their expertise on how to assess the quality of the measurement programs themselves, and what tool sets are available for deployment in industry.

Software organizations of all sizes must acquire such **know-how** about the design of software measurement programs and how to deploy them for the benefit of their stakeholders. In this book they will find effective strategies for improving quantitative software management, along with numerous industry examples for the application of best practices in designing and deploying software measurement programs.

The recommendations offered here will help software organizations design and implement efficient software measurement programs as a basis for decision-making.

Montreal, QC, Canada Alain Abran
January 2018

Preface

When a scientist and an engineer measure something, they focus on two different things. The scientist focuses on the ability of the measurement to quantify the measured thing (called the *measurand*). The engineer, however, focuses on finding the right qualities of measurement given the designed system (e.g. correctness), the quality of use of the system (e.g. ease of use) and then the efficiency of the measurement process. It happens all too often that these two focuses, seemingly contradictory, lead to a lack of consistency between the academic and industrial views on software measurement. In this book, we argue that both focuses are necessary and complementary. We argue that we can focus on the exact quantification and efficiency at the same time. We, finally, argue that industrial measurement programs need to combine both accuracy and efficiency to provide software development teams, project managers, product managers and quality manager with the right measurement methods, tools and instruments.

We, the authors of this book, come from academia and industry, where we worked together for the past 12 years. Our work has always been driven by the need to satisfy information needs of our stakeholders—product owners in the beginning and recently empowered software development teams. We have worked with both small and large software development organizations, as both researchers and as measurement engineers, measurement program leaders and even teachers. We have seen a number of measurement initiatives fail and equally many succeed, all too often depending on small things. These small things could be automation of measurement processes, combining of objective measures with subjective judgement or continuous evolution of key performance indicators in organizations.

Based on our experience we set off to organize and document our experiences. The result of this documentation is the book which you, dear reader, hold in your hand now. We wrote the book to help you define, implement, deploy and maintain a company-wide measurement program—a set of measures, indicators and roles which are built around the concept of measurement systems.

We organize the book as a gradual progression from theories of measurement (yes, you need theories to be successful!) to practical, organizational, aspects of

maintaining measurement systems (yes, you need the practical side to understand how to be successful). Whenever possible, we provide anecdotes and examples from our experiences to help you realize what is important and what the common pitfalls are. We start the book with Chap. 1, where we introduce the need for measurement and how modern science looks upon major concepts like measurability; we also look at major differences between the academic and industrial view on measurement. In Chap. 2 we dive into details of the theoretical foundation of measurement—metrology and standards like ISO/IEC 15939 (Software and Systems Engineering—measurement processes). In Chap. 3, we introduce the main characters of this book—a measurement system and a measurement program. These characters stay with us for the rest of the book and guide us in the next chapter—Chap. 4. In Chap. 4, we explore the question *How do I know that my measurement program is of a good quality?* In particular, we describe the reasoning which we experienced throughout our collaboration—what makes some measurement programs successful and what makes other programs fail. We uncover such jack-of-all-trades tricks like how to find a good stakeholder and how to reduce the cost of measurement in an organization. In Chap. 5, we uncover other tricks related to the tooling for quantification (i.e. measurement instruments), data processing and analysis, and finally visualization. Once we have provided the examples of tooling, we can share some of the experiences on which measures/indicators work and which do not (and why!). Chapter 7 is no longer about our experiences, but about what we see as the future of software measurement—machine learning and flexible counting algorithms. From what we see today, it is these areas which will shape the field of engineering software systems and therefore we have to understand the principles of how a machine (an algorithm) "understands" numbers. In Chap. 8 we dive into the topic of deployment and maintenance of measurement programs in large software organizations. Based on our experiences from introducing over 40,000 measurement systems in more than a dozen companies, we share the tips and tricks on how to do it in the most painless way possible.

We hope that you will enjoy reading this book and that it will help you to understand how to become a successful scholar and practitioner in the area of software measurement!

Gothenburg, Sweden
January 2018

Miroslaw Staron
Wilhelm Meding

Acknowledgements

First and foremost, we would like to thank our industrial partners and colleagues, who have contributed to our work and learning over many years.

In particular, we are grateful to Micael Caiman from Ericsson, who has actively supported and believed in our work for over a decade. This book would not exist if it were not for his support and encouragement.

We thank our colleagues from Software Center (www.software-center.se), who helped us to understand the complexity of industrial work on many levels, in particular: Prof. Jan Bosch, Dr. Anna Sandberg, Kent Niesel, Anders Henriksson, Christoffer Höglund, Ola Söder, Magnus Bäck, Anders Caspar, Gert Frost, Sajed Miremari, Peter Ericsson and many, many more. There would be no measurement theme and measurement projects if it were not for them.

We are very grateful to Tom Gilb, who has provided us with ideas to improve the book and contributed directly to it, in particular, for letting us see a bigger perspective of measurement programs and for helping us to address a wider audience.

We thank our colleagues from the department of Computer Science and Engineering at Chalmers/University of Gothenburg, who contributed through discussions about measurement and metrics over the years.

We are also grateful to our many colleagues from Ericsson, who supported us over the years. In particular we thank Karl-Johan Killius for his decisive support, as well as Jonas Bjurhede, Tommy Davidsson and Per Sundvall. Without their commitment, engagement and extraordinary technical skills there would have been no state-of-the-art, fully automated infrastructure.

We would like to thank our publisher, Ralf Gerstner, from Springer, who supported us in writing and constantly provided us with invaluable advice.

Finally, we are indebted to our families, for their unconditional love, support and understanding. There would be no book, no research and no fun, without them in our lives.

Contents

1 Introduction 1
- 1.1 Academic and Industrial View on Software Measurement 1
- 1.2 What a Measurement Program Is: A Brief Introduction 2
- 1.3 What a Successful Measurement Program Is 3
- 1.4 A Brief History of Software Measurements 4
- 1.5 Measures in Decision Processes 6
  - 1.5.1 Decisions-Measures Dependency Model 6
  - 1.5.2 Effective and Efficient Measures for Decision Formulation 8
- 1.6 Content of This Book 8
  - 1.6.1 Chapter 2: Foundations 9
  - 1.6.2 Chapter 3: Measurement Programs 11
  - 1.6.3 Chapter 4: Quality of Measurement Programs 11
  - 1.6.4 Chapter 5: Tooling in Measurement Programs 12
  - 1.6.5 Chapter 6: Measures in Measurement Programs 12
  - 1.6.6 Chapter 7: New Techniques 13
  - 1.6.7 Chapter 8: Maintaining and Evolving Measurement Programs 14
  - 1.6.8 Chapter 9: Summary and New Directions 15
- 1.7 So, Let's Start 15
- 1.8 Related Reading 15

References 16

2 Fundamentals 19
- 2.1 Introduction 19
- 2.2 Software Metrology and the VIM Standard 21
  - 2.2.1 Lines of Code Measure: Defined from the Perspective of the VIM Vocabulary 22
- 2.3 Mapping of Real-World Objects to Measures 25

2.4 The Concept of Measurement Error 26
2.4.1 Impact of Measurement Error on Predictions: Standard Error of the Estimate 27
2.5 ISO 15939: Measurement Processes 28
2.5.1 Base Measures and Measurement Methods 29
2.5.2 Derived Measures and Measurement Functions 32
2.5.3 Indicators and Analysis Model 32
2.5.4 Definition of Test Progress Using the ISO/IEC 15939 Measurement Information Model 33
2.6 ISO 25000: SQuARE 34
2.7 Other Standards 37
2.8 Scientific Work Provides Closure to the Standards 37
2.8.1 Types and Properties of Measures 38
2.9 Measurement Systems 40
2.9.1 Architecture of a Measurement System 40
2.10 Summary 42
2.11 Further Reading 42
References 43

3 Measurement Program 47
3.1 Introduction 47
3.2 Measurement Program Model 49
3.2.1 Measurement Systems 50
3.2.2 Information Products 51
3.2.3 Measurement Experience Base 51
3.2.4 Measurement Infrastructure 52
3.2.5 Measurement Organization 53
3.2.6 Input 54
3.2.7 Output 54
3.3 Processes 56
3.3.1 Designing a Measurement Program 57
3.3.2 Deploying a Measurement Program 72
3.4 Stakeholders 77
3.4.1 What Makes a Stakeholder, a Good Stakeholder? 77
3.4.2 The Role of the Stakeholder in the Design and Deployment Processes 78
3.5 Summary 80
3.6 Further Reading 80
References 81

4 Quality of Measurement Programs 83
4.1 Introduction 83
4.2 Data and Information Quality in General 84
4.2.1 ISO/IEC 25012 Data Quality 85

4.3 Measurement Systems Represented as Data-Flow Architecture 87
4.3.1 ISO/IEC Standards 87
4.3.2 The Need for Information Quality 89
4.4 Information Quality 90
4.4.1 Fundamentals 92
4.4.2 Assessing Information Quality: Realization 92
4.5 Completeness and Predictiveness of Measurement Programs 99
4.5.1 Fundamentals 99
4.5.2 Assessing Completeness 102
4.5.3 Completeness of Measurements in the Program 105
4.6 Assessing the Measurement Program's Organization 106
4.6.1 How to Use the Tables/Checklists 107
4.6.2 Who Should Assess the Measurement Program? 108
4.6.3 Who Should Be Assessed? 108
4.7 Summary 111
4.8 Further Reading 112
References 113

5 Tooling in Measurement Programs 117
5.1 Introduction 117
5.2 Measurement Tools and Instruments 118
5.3 Measurement Instruments 119
5.3.1 Examples of Measurement Instruments 120
5.4 Measurement Tools 123
5.4.1 Source Monitor: A Simplistic Measurement Tool for Source Code 123
5.4.2 Understand C++: An Integrated Measurement Tool and Static Analysis Tool 123
5.4.3 SonarQube: An Integrated Measurement Analytics Tool 127
5.4.4 Eclipse Metric Tool: A Measurement Tool Integrated in the Eclipse Software Development IDE 130
5.4.5 Microsoft Visual Studio Profiling Tools: Collecting Dynamic Runtime Measures 131
5.4.6 Measurement Instruments and Measurement Tools Are Only the Beginning 132
5.5 Tools from the Perspective of a Data Flow 132
5.6 Source Systems and Databases 134
5.6.1 Software Build Tools: Jenkins 134
5.6.2 Requirement Tools: IBM Rational Doors 136
5.6.3 Software Repositories 138
5.6.4 Defect Databases 140
5.7 Data Aquisition 143
5.8 Back End and Statistical Tools 145
5.9 Front End and Business Intelligence 147

5.10 Front End: Dashboards—A Popular Way of Visualizing Indicators in Measurement Programs 150
5.10.1 Examples of Charts Used in Dashboards 151
5.11 Open Data Sources 157
5.12 Summary 160
5.13 Further Reading 160
References 162

6 Examples of Measures in Measurement Systems 165
6.1 Introduction 165
6.2 Examples of Measures in Measurement Programs 166
6.2.1 Release Readiness 167
6.2.2 Forecasting Defect Backlog 170
6.2.3 Identifying and Monitoring Bottlenecks 172
6.2.4 Internal Quality Measures for Software Architects 175
6.2.5 Implicit Architectural Dependencies 179
6.2.6 Monitoring Progress of Software Development Teams 183
6.2.7 Customer Defect Inflow 184
6.3 Measurement Areas 186
6.3.1 Measurement Areas: Mind Map 187
6.3.2 Measurement Sub-areas: Selected Details 189
6.4 Summary 198
6.5 Further Reading 199
References 199

7 New Techniques 203
7.1 Introduction 203
7.2 Big Data 204
7.2.1 Volume and Velocity 205
7.2.2 Value and Variety 205
7.2.3 Veracity 206
7.3 Machine Learning 208
7.3.1 Clustering 208
7.3.2 Classification 211
7.3.3 Other Machine Learning Applications in Software Measurement 217
7.4 Measurement Reference Etalons 218
7.4.1 Measurement Reference Etalons in Software Engineering 220
7.4.2 How to Define New Measurement Reference Etalons 221
7.5 Other Trends 221
References 222

8 Maintaining and Evolving Measurement Programs 225
8.1 Introduction 225
8.2 Designing Measurement Programs for Sustainability 226
8.2.1 Designing Long Lasting Information Products 226

8.3 Selecting the Right Dashboard 227
8.3.1 Examples 229
8.3.2 Designing the Measurement Infrastructure 232
8.4 Maintaining Measurement Programs Cost and Time Efficiently 233
8.4.1 Maintaining Measurement Systems 234
8.4.2 Maintaining Information Products 234
8.4.3 Maintaining the Measurement Experience Database 236
8.4.4 Maintaining the Measurement Infrastructure 236
8.4.5 Maintaining the Measurement Organization 237
8.5 Keeping Up with Technological and Organizational Evolution 238
8.5.1 Evolving Measurement Systems 239
8.5.2 Evolving Information Product 240
8.5.3 Evolving the Measurement Experience Database 241
8.5.4 Evolving the Measurement Infrastructure 241
8.5.5 Evolving the Measurement Organization 245
8.6 Summary 246
8.7 Further Reading 247
References 248

9 Summary and Future Directions 251
9.1 Introduction 251
9.2 Measurement Program: Recommendations on How/Where to Start 252
9.2.1 Metrics Team 253
9.2.2 Standards 253
9.2.3 Automation 254
9.2.4 Company/Organization Specific Cultural Aspects 255
9.2.5 Management Focus 256
9.3 Future: Autonomous AI-Based Measurement Systems 257
References 258

Chapter 1
Introduction

Abstract In this chapter we introduce the problems which are addressed by software measurement—e.g., providing quantitative insights, and we describe the possibilities which open up when we have software measurement of products, processes and enterprise in place. We discuss the possibility of quantitative fact-based management, customer data-driven development and using artificial intelligence (or machine learning) once we have a solid measurement program. Towards the end of the chapter we outline the concept of a company-wide measurement program and introduce the content of the book.

1.1 Academic and Industrial View on Software Measurement

Measurement is a great activity that engineers love to do; it makes the discipline of engineering so elegant and structured. Software engineering is no exception to that. We can program and create software, but it's really difficult if we cannot measure properly. Because, if we can't measure our activities and products, how do we know that we're on the right track in their development.

Together with the ability to process large data sets—the so-called Big Data systems—measurement has gained a strategic value for modern enterprises. Regardless, whether we consider a large software development company with distributed projects or a small agile team developing a mobile game, measurement is strategic. The large companies need the measurement to capture the market's requirements and to quantify the complexity of their large software projects (or their large number of small projects). The small software development companies need the measurement to assess whether the company's choices are right for their customers; the small companies need to show to their customers which assets they have and how to cash in on them.

Measurement is also strategic in today's enterprises because of the ability to use data for complex decision models and complex algorithms [ASH+14, ASM+14]. We can see that modern cars are using the data to autonomously drive and to provide a completely new experience for their customers. We can also see that advanced machine learning and artificial intelligence can take advantage of measurement and

M. Staron, W. Meding, *Software Development Measurement Programs*,
https://doi.org/10.1007/978-3-319-91836-5_1

provide new values for customers. One of the applications which we are used to is stock market software trading agents, which trade stock every fraction of a second based on such measurements as stock price, the price's predicted development and risk.

If you have even encountered problems on how to measure one of the following, then the book is for you:

- Quality of your software product.
- Size or complexity of your software product.
- Availability, reliability or maintainability of your piece of software.
- Performance of your software development/maintenance/management organization.
- Progress of your software project.
- Release readiness of your product.

These aspects are so common that almost all software engineers encounter them and you are not alone. Luckily, we are here to help you.

Our book has been written as a result of a long-term (over a decade long) cooperation between academia and industry [SPA11]. It started with a pragmatic need from one of our industrial partners; it started with a question: "What should we measure?" The academic answer to that question, at the time, was quite pragmatic—give us 2 weeks and we will tell you. Unfortunately, even after the decade-long research project, we still cannot answer that question. Why? Simply because the reality around us changes so fast that our answer is out-of-date already at the moment when it is given.

This "give me 2 weeks" answer shows the challenge and the need for common language between academic innovation and industrial practice. What's easy to draw on paper (theoretical solution) is often very hard to realize in practice (practical realization of the solution). Therefore, we set off to write a hands-on guide for practitioners and academics on how to establish a measurement program.

1.2 What a Measurement Program Is: A Brief Introduction

A measurement program is a socio-technical set of measurement systems and their users. The notion of a *measurement system* is crucial for this work and we introduce it properly in Chap. 2. However, for the sake of discussion, we can see a measurement system as a software program which collects data about a number of attributes (e.g. size of a program, quality) and displays it to a stakeholder (a person who has the mandate to monitor and act upon the data displayed by the measurement system).

It is exactly here where we see the first challenge—the need to simultaneously work at two levels: the technical measurement systems and the social stakeholders and their organizations.

Based on our experience, this distinction, between the technology and social aspects of measurement programs, provides an efficient framework to separate the concerns of how, what and why.

The social part of the measurement program provides the meaning for the measurement and designates its value. We could say directly that a measure without a stakeholder is a waste of an organization's scarce resources and should be abandoned as soon as possible. We could also say that the higher the value of the measure, the more strategic it is for the company.

The technical part of the measurement program provides the process of data collection and management to arrive at the measure of the "what." We see these two parts as inseparable and essential for the success of the measurement program.

The measurement processes, used in the measurement programs, apply to *values*, *qualities* and *management objectives*, for products, systems and related organizations. We need to initially analyze our product and system environment, then determine the many critical stakeholders in that environment. We then need to determine the critical needs of those many stakeholders. The "critical needs" need to be specified in a disciplined manner [Gil05], with well-defined "scales of measure" and well-defined levels of future performance levels. This value specification is a logical prerequisite for at least three things:

1. the time of delivery prioritization of the required levels of performance,
2. the clarity of input to the architecture or design phase, and
3. the selection of one or more practical measurement processes, so as to manage and confirm delivery of the planned performance levels.

The measurement processes are our best cost-effective choices for getting feedback information about the progress of improving, or failing to improve, our critical values and objectives. In advanced engineering cultures measurement processes are in fact an engineering design decision based on many requirements and constraints. Most real cultures select measurement processes by default, by culture or by ignorance.

A single product value might employ several different measurement processes during its life-cycle, for example in early stepwise deliveries, in handover to the market, and in operation. However, these processes need to be carefully planned, executed and evaluated; thus we need to understand how to assess the quality of the measurement processes and their overarching measurement programs.

1.3 What a Successful Measurement Program Is

There are many definitions of what a "successful" measurement program is in this context [SM16, SM11]. For us, a successful measurement program is such a program that optimizes the usage of resources for measurement and the value delivered to the stakeholder of the measurement program. For example, it operates using minimal resources (both human and computer), maximizes the value of

measures to their stakeholders and can be quickly adapted to the evolution of the organization and its products. This book is structured to address these items.

Measurement programs that are unsuccessful, to the contrary, are such programs as those that:

- use significant amount of resources, competing with the product development for these resources,
- require specialized competence, competing with the product development for the competence,
- provide "carved in stone" measures, making it hard to follow the evolution of company's products, processes, strategies,
- ignore the needs of their stakeholders, making the measures in the program useless and leading to parallel, unofficial measurement programs, or
- use off-the-shelf, one-size-fits-all measures to benchmark the company's specific business, leading to wrong business and technical decisions.

The view of measurement programs has changed over time. We can see that many of the modern enterprises today can be characterized by:

- focus on customer value rather than technology for its own sake [Sta12],
- rapid response to the evolving market [SMH+14],
- operating in an ecosystem-based software development environment [Bos16],
- operating in a rapidly changing markets, where disruption is common [Col01],
- relying on empowerment of their employees for innovation, development and maintenance of their products [Tes14, RGMR16],
- relying on innovation to find new market niches for new products and constantly evolve [kim],

The book is primarily for the companies that expose at least two of the above characteristics, although every company in the software business will find parts of the book valuable and important for their measurement programs.

In order to better understand the novelty of this book, let us go through the historical developments in the area of measurement programs.

1.4 A Brief History of Software Measurements

One of the first books about software measurement was Gilb's *Software Metrics* from 1976 [Gil76]. Although the area of measurement has been around before, the book opened up the area for practitioners. In particular, the book introduced such fundamental concepts as, for example: (1) all variable attributes (qualities, values and costs) can be expressed quantitatively, (2) for any defined scale of measure, it is possible to define one or more methods for practical and economical measurement (and more than one measurement process might be useful and economical at any given time).

The measures presented by Gilb, used to monitor quality, qualified the product directly. In 1979, Albrecht presented another way of measuring software—by combining multiple measures to provide more direct support for decisions [Alb79]. The method presented there laid the foundations for size measurement using function points. This area is still being actively developed with certification programs and wide adoption in industry.

In the 1980s, the area of object-orientation developed, and so did the related measures. In particular, the object-oriented measures by Chidamber and Kamerer appeared, although these were published in 1994 [CK94]. Boehm's Constructive Cost Model (CoCoMo, [B+81]) introduced an alternative to function point measurement, and further popularized the idea of measurement for estimation of project costs. In 1986, Motorola popularized their Six Sigma approach [Har94]. Six Sigma introduced many of the concepts of statistical quality control, still used today. The example of these concepts were statistical inferences, confidence intervals, and time series.

A lot happened in the area in the 1990s. The development of Team Software Process by Huphrey (TSP, [McA00]) popularized the control of quality on the team level and on the individual level. TSP popularized concepts similar to Six Sigma, but with lower granularity, balancing statistical methods, manual data collection, reporting and documenting, with the benefits of quantitative management of projects and products.

Standards like the ISO 9126 were approved and the standards in metrology were introduced into software engineering [oWM93]. These standards provided the industry with reference points in the area of measurement. The ISO/IEC 9126 provided the first standardized way of documenting measures, which "lives" in the new editions of these standards like the ISO/IEC 25000 series.

The book by Fenton, *Software Metrics: A rigorous and practical approach*, marked the importance of both theoretical and empirical measurement validation [Fen96]. The book bridged many of the practical aspects of measurement (e.g. definitions) with the academic view on measurement (e.g. validation of measures).

It was also the end of 1990s when software development companies used software measurement programs to a large extent. Almost all large companies had some sort of measurement program. The major characteristics of these programs, however, was their limitation to either quality assurance (descendants of Six Sigma) or cost estimation (descendants of function point measurement).

To widen up this view on measurement, in the 2000s, such standards as the ISO/IEC 15939 were introduced (originally approved in 2002, revised in 2007). These standards defined the way in which measurement processes should be established. This popularized the notion of measurement programs even in smaller enterprises. Together with the introduction of efficient presentation of information, dynamically on web pages, this popularized dashboards and measurement at all levels of organizations. In contrast to the centrally managed measurement programs of the 1990s, the 2000s measurement programs can be characterized by agility, team involvement and measurement of multiple properties of software products.

In 2010, the book of Abran [Abr10] updated the view on measurement by providing metrological foundations to modern software measurement. In particular, it advocated the importance of using etalons in software engineering for calibration of measures.

Currently, in the late 2010s, the measurement programs have evolved to entities which support organizations in large-scale data collection. Usually, this data comes from customers using software products (e.g. dating back to the first, large-scale, error reporting of MS Windows 95), where companies use the data to improve their products and provide new services. All modern enterprises, successful in business, have this kind of measurement programs in place.

1.5 Measures in Decision Processes

The purpose of this sub-section is to help stakeholders to understand how to use measures in an effective way in the decision process. We use the measure-decision dependency model, based on the principles from our previous work [Sta12]. The model is shown in Fig. 1.1. In the model we distinguish between measures and indicators, on the one hand, and system and strategic decisions on the other hand. The distinction between measures and indicators stresses the fact that indicators are important to trigger decisions (metrics push) whereas measures are used to monitor the execution of the decisions. The distinction between the system measures and the strategic measures reflects two types of stakeholders in modern companies—product-oriented and business-oriented.

The model is important for the companies as it allows them to filter out measures which are not related/used in any decision making processes.

1.5.1 Decisions-Measures Dependency Model

The decisions-measures dependency model contains four distinct types of measures that are closely related to formalizing and implementing decisions: two types have more influence on formalizing decisions, while two types are more important for implementing the decisions [Boe84].

The model lists measures that relate to the type of decision, i.e. product (e.g. memory consumption of a software product) or strategic (e.g. budget). The model lists also measures that relate to the type of the measure, i.e. base measures, derived measures, and indicators [SMN08].

Indicators often trigger decisions—when measures draw the attention of managers to the entity they measure (e.g. when problems occur), e.g. during decision meetings [FSHL13, SHF+13].

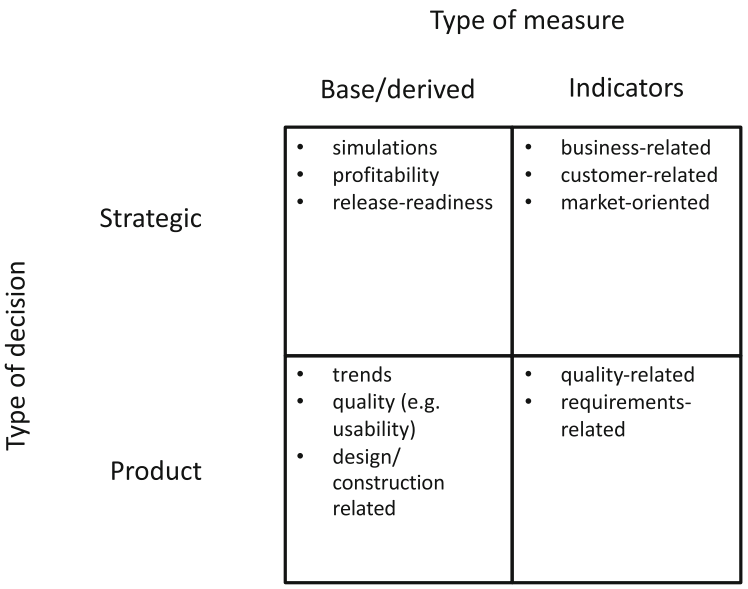

Fig. 1.1 Metrics—decisions dependency model, adapted from [Sta12]

Base and derived measures, on the other hand, are triggered by measurement processes to make a concrete decision in the organization—when managers demand collecting new measures and/or new analyses in order to take decisions [SMP12].

Strategic decisions are defined as decisions related to stakeholder value and organization of software development [Boe84]. These decisions usually impact the software development organization directly and the development product indirectly—resource reallocation might lead to re-planning of features included in a release.

Product decisions are defined as decisions related to the product internal and external properties and impact the product directly [RS05]. These decisions are often triggered by measures related to product quality.

Triggering of decisions is caused by management indicators—when indicators show "red," things happen in organizations. Usually, managers use technical indicators to formulate decisions, complementing them with simulations using managerial measures, e.g. "What if…" analyses. For example, when implementing a cost reduction program, managers need to observe the actual burn rate (compared to the simulated one) and interpolate it with current product quality trends. The technical indicators often push decisions in specific directions when decisions are formulated, although they could also trigger decisions. They have also the potential to change the course of a decision already being implemented.

While implementing the decision, base/derived technical and managerial measures are of highest interest for companies or organizations. The differentiating aspect between the managerial and technical measures is the need for monitoring trends of technical base/derived measures.

1.5.2 *Effective and Efficient Measures for Decision Formulation*

In the dependency model, the most effective measures for decision formulation are indicators, both for strategic decisions and for product ones. These indicators should have the following characteristics:

- Clear analysis model (decision criteria for interpretation: whether the indicator shows a positive—"green"—or a negative—"red"—status).
- Compact presentation and information provision, e.g. in the form of mobile apps and MS Vista gadgets.
- Long feedback loop for the managerial indicators: ca. 1–2 years.
- Short feedback loop for the technical indicators: ca. 0–20 weeks.

This book supports both the technical and managerial decision-making processes. Managers can read about how to use measures and how to support their organizations in setting up measurement systems. Architects and designers can read about how to cost-efficiently set up measurement systems and how to collect the data needed for making decisions.

Regardless of their position in the organization, all practitioners can read about the pragmatics of how to work with measurement systems, maximize the benefits of measurement and minimize its costs.

1.6 Content of This Book

The goal of this book is to popularize the structured, standardized and accurate use of software measurement at all levels of modern software development companies. We could have a subtitle—"Hands-on instructions on how to design, deploy and maintain software measurement programs, in modern software development companies," and it would capture the focus of this book and its intentions in general. In particular, we focus on the following aspects:

1. sound scientific foundations—the measurement program should include state-of-the-art measures and the latest developments in the area of product sizing, quality management and technology adoption,
2. cost-efficiency—the measurement program should cost as little as possible,
3. standardization—the measurement program should be based on the latest standards in the area of measurement, to maximize the portability and dissemination of the program,
4. value-maximization—the measurement program should maximize the value for its stakeholders, the stakeholders' companies and their customers,
5. flexibility—the measurement program should evolve at the pace of the evolution of its companies and enterprises,

6. combining organizational and technical aspects—the measurement program should maximize the stakeholders' insight using modern technology, and
7. seamless technology integration—the measurement program should maximize the adoption of machine learning technologies to reduce the burden of measurement for the enterprise.

The book is written to support these aspects and in general to increase the automation in measurement programs. It supports the companies in their journey from manual reporting to automated decision support, i.e. from Figs. 1.2 to 1.3

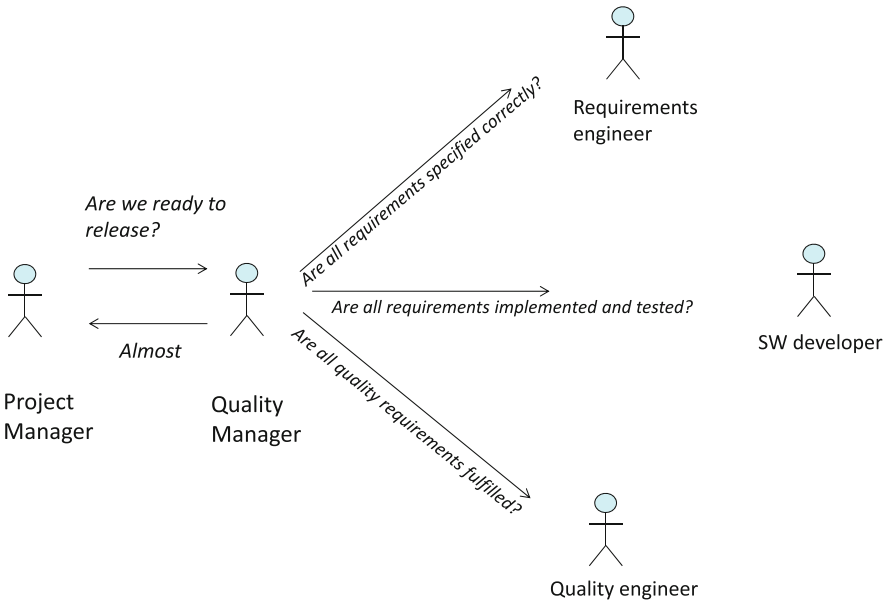

Fig. 1.2 Building opinion based on manual data collection involves multiple roles and is effort-intensive

The content of the book supports this by combining academic research and industrial practice.

1.6.1 Chapter 2: Foundations

We start by describing the theories and standards important for the modern measurement programs. We overview such standards as ISO/IEC 25000 and ISO/IEC 15939, which are needed to fill the content of the measurement programs and for structuring it.

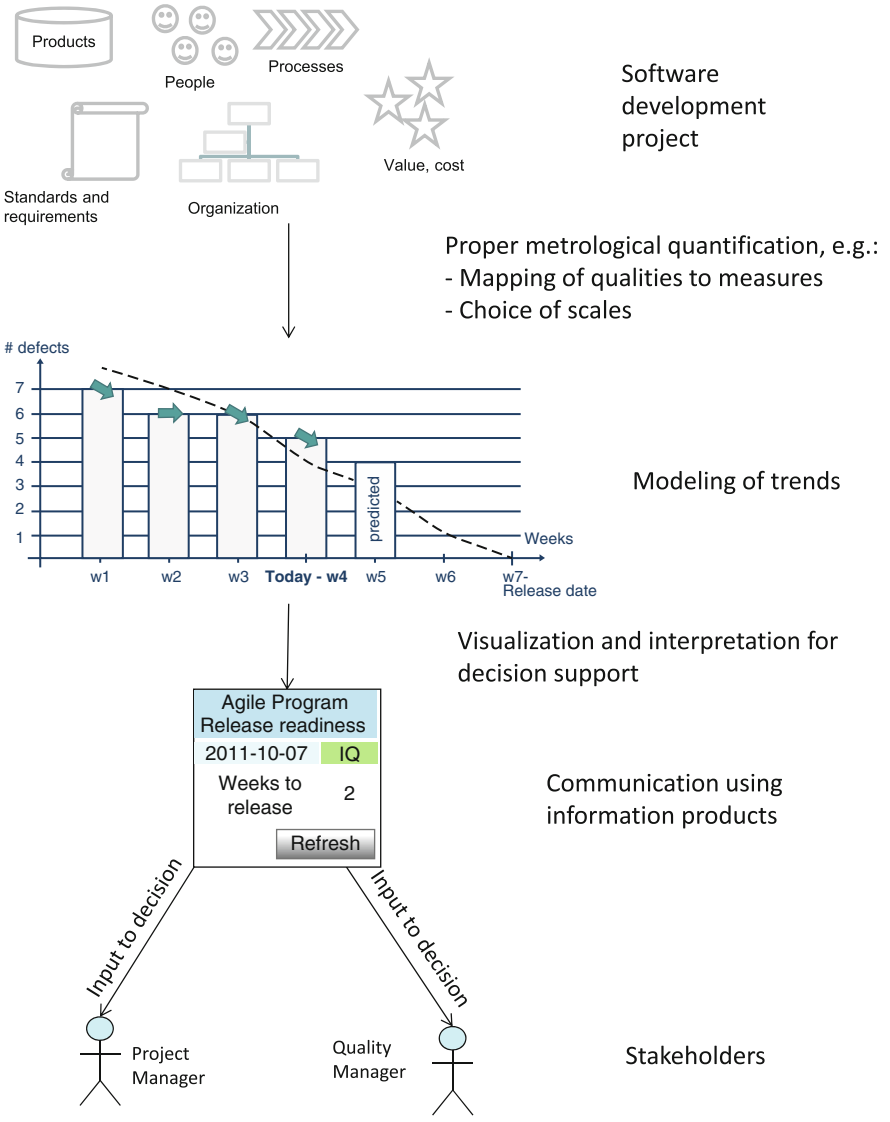

Fig. 1.3 Automation provides insight and increases efficiency of the organization

We describe also the ways in which measures should be defined and documented, based on templates from these standards. These descriptions provide the necessary foundations for further exploration of the more advanced concepts of metrology. These more advanced aspects include calibration of measurement instruments and empirical measure validation.

In Chap. 2 we provide the vocabulary which we use throughout the book and we focus, as mentioned, on foundations.

1.6.2 Chapter 3: Measurement Programs

The natural next step is to continue with the description of measurement programs in modern enterprises. In Chap. 3 we present the measurement program model, which describes elements of a measurement program—see Fig. 1.4.

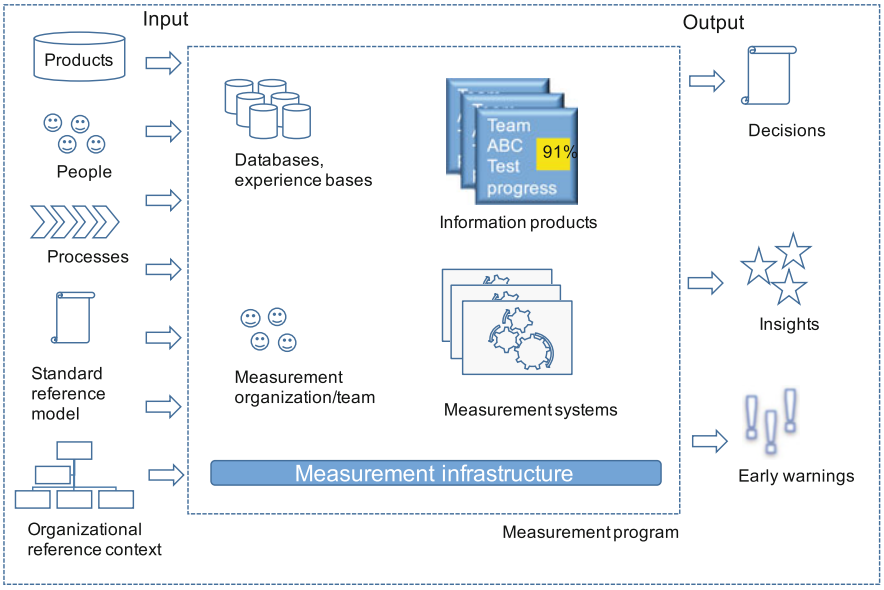

Fig. 1.4 Conceptual model of a measurement program

In the chapter we provide checklists for designing each of these elements, e.g. measurement information products.

Once we know the structure and the first-order entities of the measurement programs, we can continue to discuss the quality of measurement programs.

1.6.3 Chapter 4: Quality of Measurement Programs

The aim of Chap. 4 is to provide the foundations of what a good measurement program is. Based on the measurement program model presented in Chap. 3, we provide four views on quality of the measurement programs:

1. data quality—where we focus on the quality of the data used in measurement systems,