SOFTWARE METRICS
AND SOFTWARE METROLOGY

Alain Abran
SOFTWARE METRICS AND SOFTWARE METROLOGY
Press Operating Committee

Chair
Linda Shafer
former Director, Software Quality Institute
The University of Texas at Austin

Editor-in-Chief
Alan Clements
Professor
University of Teesside

Board Members
Mark J. Christensen, Independent Consultant
James W. Cortada, IBM Institute for Business Value
Richard E. (Dick) Fairley, Founder and Principal Associate, Software Engineering Management Associates (SEMA)
Phillip Laplante, Professor of Software Engineering, Penn State University
Evan Butterfield, Director of Products and Services
Kate Guillemette, Product Development Editor, CS Press

IEEE Computer Society Publications
The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available from most retail outlets. Visit the CS Store at http://computer.org/store for a list of products.

IEEE Computer Society / Wiley Partnership
The IEEE Computer Society and Wiley partnership allows the CS Press authored book program to produce a number of exciting new titles in areas of computer science, computing and networking with a special focus on software engineering. IEEE Computer Society members continue to receive a 15% discount on these titles when purchased through Wiley or at wiley.com/ieeecs

To submit questions about the program or send proposals please e-mail kguillemette@computer.org or write to Books, IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1-714-816-2169.

Additional information regarding the Computer Society authored book program can also be accessed from our web site at http://computer.org/cspress.
CONTENTS

Foreword vii
Preface ix
Acknowledgments xvii
About the Author xix

PART 1: Key Concepts for the Design of Software Measures 1

1 INTRODUCTION 3
2 FROM MEASUREMENT METHODS TO QUANTITATIVE MODELS: A MEASUREMENT CONTEXT MODEL 17
3 METROLOGY AND QUALITY CRITERIA IN SOFTWARE MEASUREMENT 47
4 QUANTIFICATION AND MEASUREMENT ARE NOT THE SAME! 67
5 THE DESIGN OF SOFTWARE MEASUREMENT METHODS 99

PART 2: Some Popular Software Measures: How Good Are They? 129

6 CYCLOMATIC COMPLEXITY NUMBER: ANALYSIS OF ITS DESIGN 131
7 HALSTEAD’S METRICS: ANALYSIS OF THEIR DESIGNS 145
8 FUNCTION POINTS: ANALYSIS OF THEIR DESIGN 161
9 USE CASE POINTS: ANALYSIS OF THEIR DESIGN 191
10 ISO 9126: ANALYSIS OF QUALITY MODELS AND MEASURES 205
PART 3: The Design of COSMIC – ISO 19761

11 COSMIC: DESIGN OF AN INITIAL PROTOTYPE

12 COSMIC: SCALING UP AND INDUSTRIALIZATION

PART 4: Other Issues in the Design of Software Measures

13 CONVERTIBILITY ACROSS MEASUREMENT METHODS

14 DESIGN OF STANDARD ETALONS: THE NEXT FRONTIER IN SOFTWARE MEASUREMENT

Appendix A: List of Acronyms

Appendix B: Glossary of Terms in Software Measurement

Appendix C: References

Index
Software organizations must respond to increasingly demanding customers in a globally competitive market and must implement best industry practices. With services and products available from vendors the world over, customers are insisting that their software systems be of high quality and with support services that challenge those of the competition while costing as little as possible.

To satisfy these demands, software organizations must have the ability to develop and maintain software to meet the customer’s needs, and it must have access to software that support the company’s business processes.

• How do you know and how do you objectively demonstrate to your customers that your software organization is performing at the top of the industry?

• Can you leverage this knowledge to develop estimation skills as a competitive advantage?

Benchmarking and estimation is based on measurements. There is a tremendous need for software measures to support software performance measurement, benchmarking and software project estimation, even more so when software is contracted out to third party suppliers.

There is currently available a large number of software measures and quantitative models proposed to the practitioners’ community for estimating software projects and measuring the quality of the software delivered. For instance, there are hundreds of measures proposed for software quality, software complexity, objects oriented as well as an impressive number of estimation models.

But …

• How many software organizations today have in place software measurement programs and use these measures and models as a basis for decision-making?

There must be then something at work that impairs the use of quantitative data for decision making in software-base organizations.

• What is it?
Within the software measurement community that has produced this large inventory of measures and quantitative models, there is a presumption that the lack of use of software measures in industry is caused by the practitioners’ and managers’ resistance to change.

This book is based on a different analysis and understanding of this lack of use of software measures by industry: this chasm comes from a lack of credibility in the practitioners communities, and this lack of credibility comes from the immaturity and unreliability of the measures themselves proposed to date to the industry.

Up until recently, software ‘metrics’ have been most often proposed as the quantitative tools of choice in software engineering, and the analysis of these had been most often discussed from the perspective referred to as ‘measurement theory’.

However, in other disciplines, it is the domain of knowledge referred to as ‘metrology’ that is the foundation for the development and use of measurement instruments and measurement processes.

In this book, we use as a foundation the sets of measurement concepts documented in the ISO VIM (International Vocabulary of Basic and General Terms in Metrology) to document and compare the state of maturity of measures in software with respect to classic domains of science and engineering.

- This helps in particular to document practical aspects with respect to the current design of software measures and to identify the strengths and weaknesses of their own design as measures.

What was still missing is the know-how about how to correctly design software measures, and how to recognize if a software measure is well designed, and worth using as a basis for decision-making. This book focuses precisely on these two issues.

It is up to you:

- to acquire such know-how about the design of a software measure and
- to run with it for the benefit of your organization.
A book on the design of software measures must be suited to software engineers, both practitioners and researchers.

This book presents a perspective on software measurement that, on the one hand, is new in software engineering and, on the other hand, is fairly classical in most domains of sciences, engineering, and even in all areas of business.

Here, we share years of experience in the design of software measures for their successful use as decision making tools by software managers.

Because measurement is a fundamental engineering concept, software organizations of all sizes can use this book, and managers will find in it effective strategies for improving software management, along with numerous illustrative examples.

Applying the best practices in software measurement will ensure that software engineers and managers are equipped to respond to the most demanding customers, feel supported by senior executives and are proud to be part of the software team.

In addition, this book introduces many of the theoretical concepts and references needed by professionals, managers and students to help them understand the fundamentals of the identification and evaluation of software development and maintenance processes, and of improvements to them.

This book is intended for those developing, maintaining and managing software as well as for those in software process improvements.

**STRUCTURE AND ORGANIZATION OF THIS BOOK**

This book is organized into four (4) parts and fourteen (14) chapters.

**Part 1: Key Concepts for the Design of Software Measures**

A number of the software measures proposed to the industry have deficiencies severe enough to make some of them useless to practitioners. Part 1 presents in
chapters one through five the key concepts in measurement that are necessary to recognize whether the design of a software measure is sufficiently strong to be meaningful in practice. Part 1 introduces, as well, the measurement terminology that is common in most fields of science and engineering; that is, of the metrology and related ISO standards on software measurement.

Chapter 1: Introduction.
This chapter presents the current level of maturity of software measurement within the software engineering discipline.

This chapter presents a model to understand the key concepts of software measurement as well as the measurement terminology that is consistent with measurement in all disciplines. This chapter also discusses the process necessary to design a software measurement method.

Chapter 3: Metrology and Quality Criteria in Software Measurement.
This chapter presents the set of classical concepts in metrology, and presents various definitions and quality criteria in classical measurement.

Chapter 4: Quantification and Measurement are not the Same.
This chapter presents some of the differences between quantification and measurement, and establishes a parallel with the ISO 15939 Measurement Information Model.

Chapter 5: The Design of Software Measurement Methods.
This chapter presents the key concepts and steps required to design and evaluate software measurement methods, including defining the measurement principle in software measurement up to post-design activities.

Part 2: Some Popular Software Measures: How Good Are They?
Some software measures are currently popular in the industry, often because they are easy to collect or because they appear to take into account a large number of the practitioners concerns. However, in software measurement, being popular and widely quoted is not synonym to being good. Part 2 uses in chapters six through ten the criteria from Part 1 to illustrate some of the major weaknesses in the design of a few of the software measures that are either widely used or widely quoted in the software industry.

Chapter 6: Cyclomatic Complexity Number: Analysis of its Design
Chapter 7: Haslled’s Metrics: Analysis of their Designs
Chapter 8: Function Points: Analysis of their Design.
Chapter 9: Use Case Points: Analysis of their Design.
Part 3: The Design of COSMIC—ISO 10761

Part 3 illustrates in chapters eleven and twelve how the lessons learned from the analysis of the key concepts for the design of a software measure have been put into practice to design a software measurement method conformant to the ISO criteria for a measurement method of the functional size of the software, that is the COSMIC—ISO 19761. Part 3 focuses on the design process rather than on the details of this specific measurement method.

Chapter 11: COSMIC: Design of an Initial Prototype.  
This chapter illustrates how this software measure of the functional size of software for real-time and embedded software was designed in response to an industry need. It describes in particular the process used to design the initial prototype of COSMIC, its field trials and its initial deployment.

Chapter 12: COSMIC—Scaling up and Industrialization.  
This chapter illustrates the additional effort to scale up COSMIC to increase its international acceptance and to bring it to be adopted as an international standard: ISO 19761. The key concepts of the COSMIC measurement method are also presented in this chapter.

Part 4: Other Issues in the Design of Software Measures

Part 4 illustrates in chapters thirteen and fourteen some additional issues that are traditional in measurement in day-to-day life, but that have not yet been seriously addressed in software measurement. Two specific examples are presented: convertibility across measurement design and measurement standard etalons.

Chapter 13: Convertibility across Measurement Methods  
While numerous software measures are proposed for the same attributes, there is a scarcity of convertibility studies across alternative ways of measuring. This chapter presents a convertibility analysis across two functional size measurement methods: IFPUG Function Points and COSMIC Function Points.

Chapter 14: Design of Standard Etalons: The Next Frontier in Software Measurement  
While measurement in science relies on well established standard etalons (such as for the meter and kilograms) to ensure the correctness and consistency of measurement results across contexts and countries, not a single standard etalon has yet been established for measuring software. This chapter looks at this next frontier in software measurement and reports on an initial attempt to design a first draft of a standard etalon for a referenced set of software requirements.
This book also contains three appendices:

**Appendix A:** List of Acronyms  
**Appendix B:** Glossary of Terms in Software Measurement.  
**Appendix C:** References

Additional material to complement the information contained in this book can be found at http://profs.logti.etsmtl.ca/aabran

**If you are a software manager**, you should:

<table>
<thead>
<tr>
<th>Part of the book</th>
<th>Read chapters:</th>
<th>Why?</th>
<th>What to do with it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 &amp; 2</td>
<td>Not all software measures have strong designs. These two chapters explain the quality criteria that should be expected from software measures.</td>
<td>Quality criteria your staff should look for when analyzing a software measure as a basis for decision making.</td>
</tr>
<tr>
<td>3</td>
<td>Chapter 3 on Metrology: through a quick reading of this chapter, managers will get an understanding of why software measures are still far way from the maturity of other measures in science and engineering.</td>
<td>For information only.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>This chapter positions the key ISO standard on software measurement—ISO 15939—and clarifies the differences between measurement and quantitative models</td>
<td>For a better understanding of the subtleties in measurement and quantitative analysis</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>One of</strong> chapters 6 to 10</td>
<td>Select the chapter to read on the basis of which one of these measures are in use in your organization</td>
<td>For the measures in use in your organization, ask your software engineer to document the impact of the weaknesses identified in this book on the quality of the measures and models you use for taking decisions.</td>
</tr>
</tbody>
</table>
### Part of the book

<table>
<thead>
<tr>
<th>Part of the book</th>
<th>Read chapters</th>
<th>Why?</th>
<th>What to do with it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Chapter 12</td>
<td>For an example of a software measurement method that has been accepted as an ISO standard.</td>
<td>If your organization is using software measures that have not yet been standardized, you must be careful.</td>
</tr>
<tr>
<td>4</td>
<td>Chapter 13</td>
<td>To get an example of convertibility across two distinct measures of software.</td>
<td>If your organization is mixing together numbers from different measurement designs, you should get your software engineers to document the impact this has on your quantitative models for decision making.</td>
</tr>
</tbody>
</table>

### If you are a software engineering practitioner or a software quality analyst

Using or planning to use existing software measures you should:

<table>
<thead>
<tr>
<th>Part of the book</th>
<th>Read chapters</th>
<th>Why?</th>
<th>What to do with it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 to 5</td>
<td>These chapters explain the quality criteria that should be expected from software measures. These chapters teach you what to look for when analyzing a software measurement method.</td>
<td>You should document the impact of the weaknesses identified in this book on the quality of the measures and models used in your organization for making decisions.</td>
</tr>
<tr>
<td>2</td>
<td>One of the chapters 6 to 10</td>
<td>For examples of major weaknesses in some of the popular measures proposed to the software industry.</td>
<td>If you are using one of these measures, you should review your actual interpretation and usage of the numbers you are getting out. If you are using other software measures, you should ask for similar analyses.</td>
</tr>
<tr>
<td>3</td>
<td>Chapter 12</td>
<td>Same as for managers</td>
<td>Same as for managers</td>
</tr>
<tr>
<td>4</td>
<td>Chapter 13</td>
<td>Same as for managers</td>
<td>Same as for managers</td>
</tr>
</tbody>
</table>
If you are in software process improvement or a researcher planning to analyze existing software measures or to design new software measures or if you are taking an undergraduate or graduate course on software measurement you should:

<table>
<thead>
<tr>
<th>Part of the book</th>
<th>Read chapters:</th>
<th>Why?</th>
<th>What to do with it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>All chapters from 1 to 4</td>
<td>Same as in previous tables above and an in-depth discussion on the design of software measurement methods in Chapter 5.</td>
<td>Same as in previous tables. Use the design process described in chapters 2, 4 and 5, when you have to design a new measurement method for software or when you have to correct the design of an existing one.</td>
</tr>
<tr>
<td>2</td>
<td>All chapters from 5 to 10</td>
<td>For a good coverage of the analyses documented to date on the design of some of the popular measurement methods.</td>
<td>If you have to evaluate other software measures, these chapters will guide you on how to do such evaluations. If you have to design new software measures, these chapters will also teach you what mistakes to avoid.</td>
</tr>
<tr>
<td>3</td>
<td>Both chapters 11 and 12</td>
<td>For understanding the stages of developing a software measurement method, from its initial design up to its highest recognition in the field of measurement that is: an International Standard.</td>
<td>Use a process similar to the one in Chapter 11 for introducing strengths into the design of new measurement methods for software. Use a process similar to the one in Chapter 12 for adding strengths into the design of existing measurement methods for software.</td>
</tr>
<tr>
<td>4</td>
<td>Both chapters 13 and 14</td>
<td>For an understanding of two critical issues in software measurement: using distinct measurement methods for the same attributes and the lack of standard etalon in software measurement.</td>
<td>These issues still have to be tackled by researchers for the majority of measures proposed to practitioners.</td>
</tr>
<tr>
<td></td>
<td>And All Advanced Readings sections</td>
<td>For an in-depth understanding of the issues presented in each chapter.</td>
<td>For detailed examples of concepts introduced in the various chapters.</td>
</tr>
</tbody>
</table>
This book is not about:

- A compendium of all software measures:
  - The purpose of this book is not to present an exhaustive list of measures of any type, or of a specific type (for instance on OO metrics).
  - There exists already on the market a number of books presenting inventories of alternative measures, as well as hundreds of research papers on emerging designs, which at this stage would still be fairly immature.
- A compendium of software estimation models:
  - This book does not list or discuss any of the estimation models for software.
  - For instance, COCOMO [Boehm 1981, 2000] is an ‘estimation model’ which attempts to predict the relationships across a large number of factors. COCOMO is not about measurement but a lot more about experimentation (as in science) to build prediction models. COCOMO, for instance, should be used and evaluated as an estimation model. This will be discussed in another book looking into the design and evaluation of estimation models.
- A compendium of analyses of all software measures:
  - This book presents from chapters six through ten analyses that have already been carried out in research and published at a number of international conferences.
  - A large number of software metrics, such as the ones in (or derived from) Chidamber & Kemerer metrics suite [Chidamber 1993], has not yet been analyzed from a metrology perspective. The analysis from a metrology perspective of these other measures still has to be done.

**COSMIC Function Points**

The COSMIC Function Points have been adopted in 2003 as an international standard—ISO 19761—for measuring the functional size of software. Having been designed to meet metrology criteria, COSMIC Function Points are at times used in this book to illustrate a number of measurement concepts. For more details on the design of COSMIC Function Points, see Section 5 of Chapter 12.
ACKNOWLEDGMENTS

A number of collaborators, including colleagues in industry and university as well as PhD students, have helped me over the years improve my understanding of many of the concepts presented in this book, in particular:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Co-Contributor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2: From Measurement Methods to Quantitative Models: A Measurement Context Model</td>
<td>Dr. Jean-Philippe Jacquet (France)</td>
</tr>
<tr>
<td>3: Metrology and Quality Criteria in Software Measurement</td>
<td>Dr. Asma Sellami—University of Sfax (Tunisia)</td>
</tr>
<tr>
<td>4: Quantification and Measurement are not the Same.</td>
<td>Dr. Jean-Marc Desharnais—Ecole de technologie supérieure (Canada) &amp; Bogaziçi University (Turkey)</td>
</tr>
<tr>
<td>5: The Design of a Software Measurement Method</td>
<td>Dr. Naji Habra—Facultés Universitaires Notre-Dame de la Paix—FUNDP, Namur (Belgium)</td>
</tr>
<tr>
<td>6: Cyclomatic Complexity Number: Analysis of its Design</td>
<td>Dr. Naji Habra—FUNDP (Belgium)</td>
</tr>
<tr>
<td>7: Halstead’s Metrics: Analysis of their Designs</td>
<td>Dr. Rafa Al-Qutaish—Alain University of Science and Technology, Abu Dhabi Campus, United Arab Emirates</td>
</tr>
<tr>
<td>8: Function Points: Analysis of their Design</td>
<td>Joost Ouwerkerk—Expedia (Canada)</td>
</tr>
<tr>
<td>9: Use Case Points: Analysis of their design</td>
<td>Dr. Rafa Al-Qutaish—Alain University of Science and Technology, Abu Dhabi Campus, United Arab Emirates</td>
</tr>
<tr>
<td>10: ISO 9126: Analysis of its Quality Models and Measures</td>
<td>D. St-Pierre, Dr. Desharnais, Dr. P. Bourque and M. Maya (École de technologie supérieure—University of Québec—Canada)</td>
</tr>
<tr>
<td>11: COSMIC—Design of an Initial Prototype</td>
<td></td>
</tr>
</tbody>
</table>
Above all, this book is dedicated to all those who provided me with feedback and insights on software measures over the years and who are contributing, each in his or her own way, to the improvement of software measures as a foundation for sound, quantitatively-based decision making.
Dr. Alain Abran is a professor and the director of the research group in Software Engineering Management at the École de Technologie Supérieure (ETS)—Université du Québec, Montréal, Canada (www.gelog.etsmtl.ca)


Dr. Abran has more than 20 years of industry experience in information systems development and software engineering and 15 years of university teaching. He holds a PhD in electrical and computer engineering (1994) from the École Polytechnique de Montréal (Canada) and Master’s degrees in management sciences (1974) and electrical engineering (1975) from the University of Ottawa (Canada).

His research interests include software productivity and estimation models, software engineering foundations, software quality, software measurement, functional size measurement methods, software risk management and software maintenance management.

Most of his publications can be downloaded from: http://profs.logti.etsmtl.ca/aabran/Publications/index.html
PART 1

KEY CONCEPTS FOR THE DESIGN OF SOFTWARE MEASURES
INTRODUCTION

1.1. INTRODUCTION

In the field of software engineering, the term “metrics” is used in reference to multiple concepts; for example, the quantity to be measured (measurand\(^1\)), the measurement procedure, the measurement results or models of relationships across multiple measures, or measurement of the objects themselves. In the software engineering literature, the term was, up until recently, applied to:

- measurement of a concept: e.g. cyclomatic complexity [McCabe 1976],
- quality models: e.g. ISO 9126—software product quality, and

\(^1\)A measurand is defined as a particular quantity subject to measurement; the specification of a measurand may require statements about quantities such as time, temperature, and pressure [VIM 2007].
• estimation models: e.g. Halstead’s effort equation [Halstead 1977], COCOMO I and II [Boehm, 1981, 2000], Use Case Points, etc.

This has led to many curious problems, among them a proliferation of publications on metrics for concepts of interest, but with a very low rate of acceptance and use by either researchers or practitioners, as well as a lack of consensus on how to validate so many proposals.

The inventory of software metrics is at the present time so diversified and includes so many individual proposals that it is not seen as economically feasible for either the industry or the research community to investigate each of the hundreds of alternatives proposed to date (for instance, to measure software quality or software complexity).

This chapter illustrates the immaturity of both the software measures themselves and the necessity to verify the designs of these measures.

1.2. SOFTWARE MEASUREMENT: IS IT MATURE OR NOT?

The IEEE Computer Society defines software engineering as:

“(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in (1)” [IEEE 610.12]

From the IEEE definition of software engineering—see box; it is obvious that measurement is fundamental to software engineering as an engineering discipline.

But what is the status of measurement within software engineering, and how mature is the field of knowledge on software measurement?

In software engineering, the software metrics approach has, up to fairly recently, been the dominant approach to measurement in this new engineering discipline.

Over recent decades, hundreds of so-called software metrics have been proposed by researchers and practitioners alike, in both theoretical and empirical studies, for measuring software products and software processes [Boehm 2000, Chidamber 1993, Karner 1993, Halstead 1997, etc.]:

• Most of these metrics were designed based either on intuition on the part of researchers or on an empirical basis, or both, and they have most often been characterized by the ease with which some entities of the development process can be counted.

• In their analysis of some of them, researchers have most often used the concepts of measurement theory as the foundation for their analytical
investigation. However, while relevant, measurement theory deals with only a subset of the classical set of measurement concepts, and software metrics researchers, by focusing solely on measurement theory, have investigated mainly the representation conditions, the mathematical properties of the manipulation of numbers, and the proper conditions for such manipulations [Fenton 1997, Zuse 1997].

- In the scientific fields, including engineering, as well as in others, like business administration and a significant number of the social sciences, measurement is one of a number of analytical tools. Measurement in those sciences is based on a large body of knowledge built up over centuries, even millennia, which is commonly referred to as “metrology”.

In the literature on software metrics, there is almost no reference to the classical concepts of metrology in investigations into the quality of the metrics proposed to the software engineering community.

Only recently have some of the metrology-related concepts been introduced in the software engineering community, including the selection of the ISO vocabulary on metrology [VIM 2007] as the basis for measurement terminology for all future ISO standards on software measurement.

One of the peculiarities of software engineering relative to other engineering and scientific disciplines is its lack of general use of quantitative data for decision making. Symptoms of this are:

- a very limited number of accepted software measures available to practitioners and recognized as mature enough to be recognized as international standards, and
- a very small number of rigorous experimental studies (which constitute general practice in the engineering and medical fields, for example).

In mature disciplines, there is:

- a large international consensus on measures, in addition to established measurement methods and an etalon for each, and
- a significant number of measuring instruments to be used in different contexts and under various constraints, many of them certified and calibrated against internationally recognized, and traceable, etalons.

In mature disciplines, measures are also recognized as a necessary cost of doing business and of carrying on engineering activities, as well as a must for improving decision making.

In the software domain, we have none of the above, with the exception of the recent adoption of ISO standards for the measurement of the functional size of software, and some works-in-progress for the measurement of software quality.
INTRODUCTION

ISO Standards for Measuring the Functional Size of Software

— The ISO 14143-1 on the mandatory set of characteristics of software functional size (i.e. a meta-standard),
— Five (5) ISO recognized specific measurement methods to implement the quantification of these functional size characteristics: ISO 19761- COSMIC, ISO 20926-IFPUG, ISO 20968-MKII, ISO 24570-NESMA and ISO 29881-FISMA.

Note: The software functional size measurement process is not yet mature enough for there to be a single universal way of measuring software functional size.

ISO Standards for the Measurement of Software Quality

The set of models of software product quality in ISO 9126-1 constitutes an international standard.

The three catalogs of more than 250 “metrics” in Parts 2 to 4 of ISO 9126 are still only ISO technical reports: much work remains to be done to bring them up to ISO standard status.

What does this mean for software measurement?

• Many, if not most, of the software measures proposed to the industry have not been seriously analyzed, nor are they sufficiently mature.
• In contrast to other fields of science and engineering, these software measures lack the credibility to be used as a basis for decision making.
• Verification criteria for software measures should be comprehensive, carefully defined, and agreed upon.
• Designers of software measures should document how well their proposed measures meet these verification criteria.

Impact of Lack of Credibility of Software Measures

It is not until it can be demonstrated unambiguously that a proposed measure achieves a high level of measurement quality that it can be expected to reach a level of credibility in the practitioner and manager communities, and then be used in practice on a large scale.

Impact of the absence of software measure credibility: when objective and quantitative data are required for decision making in software engineering, software engineering researchers and practitioners must often design and develop their own individual software measures and measurement methods, whereas these already exist in other fields of knowledge.
1.3. SOFTWARE MEASUREMENT AS A NEW TECHNOLOGY

Technology is defined as the set of methods and materials used to achieve industrial or commercial objectives.

This definition does not limit technology to materials alone: it also includes processes and the knowledge related to them, referred to as “know-how.”

From that perspective, software measurement is a technology.

While some technologies are quite mature and widely used by practitioners in industry, others are emerging and in need of significant improvement if they are to penetrate deeply into an industrial domain.

- Mature technologies: they typically have been fine-tuned over many years and they have been adapted with a number of features and tools to fit various contexts and to facilitate their use by non experts.
- Innovative and immature technologies: they require significantly more expertise for using them in their ‘immature’ status.
- Innovative (and immature) technologies are used initially by innovators who try them, test them and invest in improving them to facilitate their use within their technical context. Innovators work at bypassing initial design weaknesses to facilitate their use and adoption by people with less expertise.

Software measurement is definitively a new technology, and, as such, it shares many of the characteristics of new technologies and as well as the constraints that must be tackled to facilitate its adoption by industry at large and by individual practitioners.

What does it take for a new technology to be adopted?

On the part of an organization:

- The new, initially unknown technology must promise enough benefits to overcome the pain of changing from a known one.
- The organization must have (or gain) the technological know-how to make it work.
- The organization must be clever enough, and enthusiastic enough, to harvest its benefits, which takes time.

On the part of an industry:

- The new technology must become integrated into the industry’s technological environment.
- It must also become integrated into the business context (which includes its legal and regulatory aspects).
- It must have been proven to work well in a large variety of application contexts (that is, the technology must be mature, or maturing rapidly).
What does it take for an industry to promote a new technology?

- The industry must recognize that there is a direction that has been proven to work in similar contexts.
- It must recognize that current practices are not good enough.
- It must also recognize that the players will not, on their own, submit to the pain of change (unless the environmental-regulatory context forces such a change).
- It must want to speed up the transition to the new technology.

What does it take for software measurement to be adopted as a new technology?

On the part of a *software organization*:

- Software measurement must promise enough benefits to overcome the pain of changing to an initially unknown technology.
- The organization must have the technological know-how in software measurement to make it work.
- The organization must be clever enough, and enthusiastic enough, to harvest the benefits, which takes time.

On the part of the *software industry*:

- Software measurement must become integrated into the technological environment of the software industry.
- It must become integrated into the business context (which includes its legal and regulatory aspects).
- Software measurement must already have been proven to work well in a large variety of contexts (that is, it must be mature as a technology, or maturing rapidly).

What does it take for an industry to promote software measurement as a new technology?

- Software measurement must have been proven to work in similar contexts.
- Current software measurement practices must be good enough.
- The industry must recognize that the players will not, by themselves, submit to the pain of change (unless the environmental-regulatory context forces such a change).
- It should want to speed up the transition to quantitative support for decision making.