PART 1

KEY CONCEPTS FOR THE DESIGN OF SOFTWARE MEASURES

Software Metrics and Software Metrology, by Alain Abran

1

INTRODUCTION

This chapter covers:

- Software measurement: is it mature or not?
- Software measurement as a new technology
- The designs of software measures must be verified
- Advanced Readings: Measurement within the Software Engineering Body of Knowledge

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].

Software Metrics and Software Metrology, by Alain Abran

- 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.

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.

The software industry has yet to resolve many of these issues, and much work remains to bring software measurement to a high enough level of quality and maturity to meet market expectations.

1.4. THE DESIGNS OF SOFTWARE MEASURES MUST BE VERIFIED

Software measurement must play an increasingly important role in software engineering if it is to truly become an engineering discipline.

- Over the past twenty years, a significant number of software metrics have been proposed for better control and greater understanding of software development practices and products.
- Unfortunately, very few of these metrics have been looked at closely from a measurement method perspective, and it is currently difficult, because of a lack of agreed-upon frameworks of verification and validation procedures, to analyze their quality.

What constitutes a valid metrics, or even a valid measurement method? How can a measurement method be validated?

Various authors have attempted to address these questions in recent years, and from different points of view (mathematical, practical, etc.)[2].

The analytical perspective proposed in this book is complementary to previous works referred to above and discusses this issue from a measurement method point of view. Furthermore, to avoid the confusion generated by inconsistent terminology in previous studies on validation, the more precise set of concepts of *verification* criteria is preferred here over *validation* criteria.

In Part 1 of this book (Chapters 2 to 5) we will refer to *specific criteria for verification*, rather then to *generic concepts related to validation* in general.

Examples of Verification Questions that Need to be Investigated

— Does the measurement method really measure the concept intended to be measured?

— Has the measurement method been internally verified, i.e. in the sense that it can be shown that it gives a proper numerical characterization of the attribute to be measured?

— Is the measurement method usable? A measurement method which is as perfect as possible from a mathematical viewpoint would not be of any interest if it could not be applied (if it were far too time-consuming, for example).

[2]See Chapter 2, Section 6.

ADVANCED READINGS: HOW MUCH MEASUREMENT SUPPORT IS RECOGNIZED WITHIN THE SOFTWARE ENGINEERING BODY OF KNOWLEDGE (SWEBOK)?

Audience

This Advanced Reading section provides to readers an overview of how software measurement fits within the discipline of software engineering, and how much support software measurement provides to software from an engineering perspective.

Software engineering has only recently reached the status of a legitimate engineering discipline and a recognized profession.

- Up to 2003, there was no generally accepted body of knowledge in this new field.
- To address the issue, the IEEE-Computer Society initiated various task forces in the 1990s, including the SWEBOK (Software Engineering Body of Knowledge) project [Abran, Moore *et al.* 2005d], to develop an international consensus on a compendium of the body of knowledge that has been developing and evolving over the past four decades and a guide to that compendium.

Importantly, the SWEBOK is not static, and the Guide to it must, of necessity, develop and evolve as software engineering matures.

The SWEBOK was established with the following five objectives:

1. Promote a consistent view of software engineering worldwide;
2. Clarify the place and set the boundary of software engineering with respect to other disciplines, such as computer science, project management, computer engineering, and mathematics;
3. Characterize the contents of the software engineering discipline;
4. Provide a topical access to this body of knowledge;
5. Provide a foundation for curriculum development, and for individual certification and licensing material.

The material that is recognized as being within the realm of software engineering is organized into the 10 Knowledge Areas (KAs) listed in Table 1.1.

The topic of measurement was the subject of one of the editorial criteria for the initial write-up of the first draft of the SWEBOK Guide, that is, that the measurement "theme" be common to all KAs.

Measurement in engineering is an integral part of all engineering KAs, and therefore had to be incorporated into the proposed breakdown of topics in each KA for software engineering.

TABLE 1.1. The 10 SWEBOK Knowledge Areas

1. Software requirements
2. Software design
3. Software construction
4. Software testing
5. Software maintenance
6. Software configuration management
7. Software engineering management
8. Software engineering process
9. Software engineering tools and methods
10. Software quality

Since a criterion for the inclusion of knowledge in the SWEBOK Guide was that it be generally accepted, it is important to ask what did, in fact, gain approval on a consensual basis with respect to measurement, and what can be learned from this consensus, or the lack of it.

The SWEBOK Definition of "Generally Accepted"

Applies to most of the projects, most of the time, and widespread consensus validates its value and effectiveness

This definition is borrowed from the Project Management Institute (PMI).

Another tool used for the development of the SWEBOK, from an engineering viewpoint, was the Vincenti [1990] classification of engineering knowledge, including, of course, "quantitative data" as a category.

On the left-hand side of Table 1.2, the six categories of engineering knowledge are listed, while the next three columns present related sub-concepts identified in specific engineering disciplines for classification purposes.

At the beginning of the SWEBOK project, it was suggested to the specialists in charge of a particular software engineering KA that they use this classification for their initial draft.

This was, of course, a challenging assignment, because:

- the discipline of software engineering was not mature enough at the time, and
- the classification could not be *directly* implemented in most of the KAs of the software engineering taxonomy.

The Vincenti classification for engineering knowledge types is very useful for understanding the depth of coverage of some engineering topics (including measurement) within each KA.

TABLE 1.2. Classification of Engineering Knowledge—Vincenti

Vincenti Categories	Related sub-concepts		
Fundamental Principles	Operational principle: how the characteristic parts of an operation fulfill their special functions in combining into an overall operation.	Normal configuration: common arrangement of the constituent parts.	To be learned deliberately. Form an essential part of *design* knowledge.
Criteria & Specifications	Specific requirements (of operational principles). Limits (across an entire technology).	To translate general, qualitative goals couched in concrete technical terms Note: the "… ilities".	Key knowledge: selection of the appropriate set of criteria.
Theoretical Tools	*Design* concepts; intellectual tools for thinking about design.	Mathematical methods & theories for design calculations. Models: combinations of measures/ parameters.	Methods of value are micro-methods, closely tailored to the tasks of developing parts which are particularly well understood.
Quantitative Data	Represented in tables and graphs.	Obtained empirically or calculated theoretically,	Descriptive or prescriptive.
Practical Considerations	Theory: often insufficient —considerations from experience and practice. Trade-offs: resulting from general knowledge about the device, its use, and its context.	Structured procedures; Knowledge derived from practice, learned from on-the-job experience. Knowledge from an expert's skills set.	Ad-hoc assumptions about a phenomenon: not formally codified, but represented by rules of thumb.
Design Instrumentalities	Know-how: ways of thinking. Procedural knowledge.	Solutions can be sought where some element of novelty is required.	Judgment skills. Knowledge on how to carry out tasks: repeatable and documented.

For instance, it can be observed that, while the term "measurement" appears by design throughout all the SWEBOK KAs (that is, it was the subject of one of the editorial criteria), neither the KA editors nor the group of reviewers has pointed to key references providing generally accepted quantitative data for any of the topics identified in each KA.

In the 2004 version of the SWEBOK Guide, there is no reference to highly credible and documented quantitative data and relevant repositories of quantitative references.

This means, for instance, that, while there are a large number of chapters and books dedicated to estimation in the software engineering literature, the raw datasets available for study often lack engineering rigor in the data collection procedures on the one hand, and, on the other, the estimation models proposed usually have both poor explanatory power and significant limitations in generalization power.

Quantitative Data in Engineering

In engineering, quantitative data does not mean raw data, but rather descriptive or prescriptive data usually derived from controlled experiments using widely recognized measurement concepts, verified measurement instruments, documented measurement protocols, and extensive testing and replication procedures to ensure both the verification of data inputs and an in-depth understanding of the phenomena under study in order to identify both the range of operations and their limitations.

Data in Software Engineering Estimation Models

In both versions of the COCOMO models [Boehm 1981, 2000], many parameters are described by linguistic values, and their influence is determined by expert opinion rather than on the basis of information from descriptive and prescriptive engineering repositories.

Significant effort still needs to be invested by model builders in terms of incorporating engineering strength into these models.

EXERCISES

1. In every organization, there is a well-staffed and well-funded accounting department. An accounting department is basically a measurement program. Why is such measurement considered as essential to any organization, from the very small to the largest?

2. In software organizations, most consider measurement programs too expensive and too time-consuming. Why?

3. Business managers and engineers thrive on measurements and quantitative data. What about software managers?

4. In all areas of an organization, staff resources willingly collect all kinds of measures for management purposes. Why are software staff so reluctant to collect software measures?
5. In various business areas, many measures are collected manually and used for taking major business decisions. Why are manually collected measures regarded with such suspicion in software organizations?
6. In engineering organizations, significant efforts are devoted to measurement? What benefits are expected?
7. In engineering organizations, significant budgets are dedicated to acquiring or developing measuring instruments. By comparison, how much is spent in software organizations on acquiring or developing automated measurement systems?
8. In engineering and in the basic sciences, a great deal of research funding goes towards conducting experiments to collect "quantifiable data". How much of the research budget in software engineering goes into collecting such data?
9. Select three of the most often quoted "software metrics". For which of these does your organization have the technological knowledge to make them work.
10. Select two software measures collected in your organization. Identify business decisions which have been based on results from data collected with these two measures.
11. What is the ultimate recognition for a measure in most scientific fields?
12. Which criteria would you use to verify that the design of a software measure is sound?
13. Can you have engineering without measurement? Can you have software engineering without measurement?
14. Give examples of international consensus on some software measures.
15. How much quantitative data in software engineering are now recognized as generally accepted? Provide supporting evidence for your answer.
16. List five software measures and explain why you would consider them either mature or immature.

TERM ASSIGNMENTS

1. If you consider software measurement to be a new technology, what are the typical hurdles that must be overcome to implement software measures in an organization and in the software industry?

2. Of the software measures you are currently using in your organization, for which of them do you currently have documented evidence that they are based on a sound foundation?

3. Measurement and quantitative data are fundamental in engineering. Measurement is also important in the Guide to the Software Engineering Body of Knowledge—SWEBOK www.swebok.org. Identify recent sets of quantitative data which you would recommend for inclusion in the next release of the SWEBOK.