

# A Pastry Cook's View on Software Measurement\*

Frank Niessink and Hans van Vliet

Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit  
De Boelelaan 1081, 1081 HV, Amsterdam, The Netherlands  
Tel: +31 20 444 7781, Fax: +31 20 444 7653  
E-mail: {F.Niessink, J.C.van.Vliet}@cs.vu.nl

## Abstract

Many frameworks for implementing software measurement exist, ranging from collections of success factors to maturity growth models. One may ask to what extent these guidelines increase the chance of a successful measurement program. To aid in answering this question, we introduce a generic process model for measurement-based improvement. We use this model as a reference model to compare a number of existing software measurement implementation frameworks. From these assessments we conclude that the guidelines given by these frameworks provide a considerable amount of support for the basic activities needed to implement measurement programs. However, we also observe that the guidelines hardly provide any guidance to guarantee successful *usage* of the measurement program.

## 1 Introduction

Briand, Differding, and Rombach [3] state that ‘Despite significant progress in the last 15 years, implementing a successful measurement program for software development is still a challenging undertaking.’ In this paper we take a closer look at different frameworks in the literature for implementing measurement programs. One may ask to what extent use of these frameworks increases the chance of a successful measurement program. Although this question is difficult to answer, we try to make a start by investigating to what extent these guidelines: (1) agree and disagree with each other, and; (2) cover different aspects of implementing software measurement.

The frameworks for implementing measurement programs we cover in this paper are quite different in nature and structure. This makes a straight comparison of the frameworks rather difficult. Therefore, we have developed an abstract representation of the measurement process, to be used as a reference model in the comparison of the different frameworks discussed in this paper. This reference model is based on the assumption that measurement itself is never a goal, but is always a means to reach some organizational goal, or to help solve an organizational problem. Hence, we also assume that measurement activities are always performed in combination with other activities needed to solve the problem or to reach the goal; i.e. improvement activities. These improvement activities change the organization, based on the results of the measurement activities. We call this combination of measurement and improvement activities ‘measurement-based improvement’. The reference model presented in section 2 models these activities.

---

\*Software Measurement – Research and Practice of Software Metrics, Reiner Dumke and Alain Abran (ed.), Deutscher Universitaetsverlag, Wiesbaden, Germany, 1998, pp. 109-126.

In section 3, we shortly describe five frameworks for implementing measurement programs from the literature. We show how each of these frameworks maps on the reference model. Section 4 then discusses the differences and similarities between the frameworks. Finally, section 5 presents our conclusions.

## 2 Modeling Measurement-based Improvement

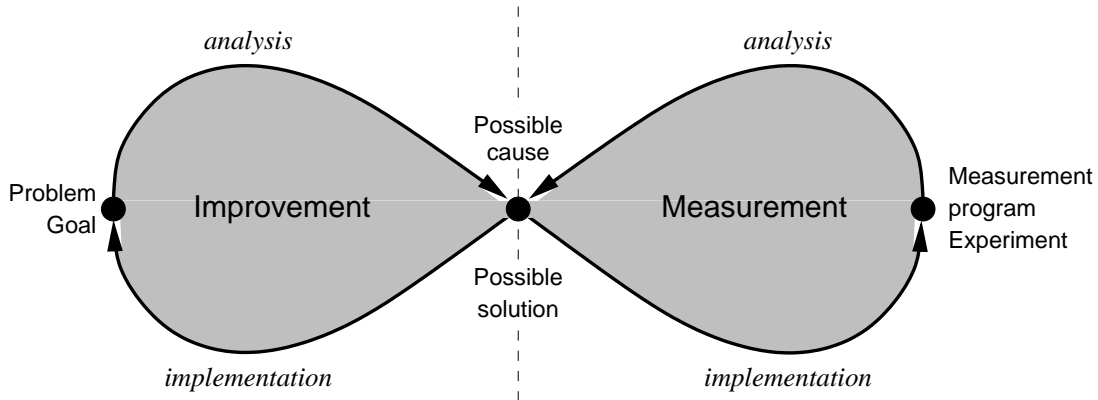


Figure 1: A generic process model for measurement-based improvement

Figure 1 displays a generic process model for measurement-based improvement. It more or less resembles a ‘pretzel’, a loaf of bread in the form of a loose knot<sup>1</sup>. The pretzel consists of two parts—the two halves, three concepts—the black dots, and four steps—the four arrows.

The cycle starts with an organizational problem or goal (left black dot). We do not assume anything about the ‘size’ of the problem or goal. A problem could only affect one developer or the whole organization, in both cases the same steps have to be passed through. The organization analyses the problem (upper left arrow), and arrives at one or more possible causes of the problem and/or possible solutions (middle dot). The analysis will generally be based on a combination of knowledge about the own organization, knowledge from literature (‘theory’), and common sense. Next, the organization has to decide whether it has sufficient knowledge to establish the cause of the problem and correct it, or to reach the stated goal. If this is the case, the organization need not traverse the right cycle. In most cases, however, the organization needs to find out which of the possible causes is the real cause of the problem, or which of the possible solutions is the best solution. Or, it may need extra information to implement the solution. To gather this information, the organization can design an experiment or set up a measurement program (lower right arrow). Executing the measurement program or experiment (right dot) results in the gathering of data, which is analyzed and related to the problem or solution at hand (upper right arrow). Finally, the organization solves the problem or reaches the goal by implementing the solutions found (lower left arrow).

Although both the preceding description and the arrows in figure 1 suggest a chronological sequence of steps, this is not necessarily the case. The arrows merely indicate causal relations. Hence,

<sup>1</sup>Of course, from a mathematical point of view, the figure looks like a lemniscate of Bernoulli. That is, the locus of points  $P$ , such that  $\text{distance}(P, p_1) \times \text{distance}(P, p_2) = (\text{distance}(p_1, p_2)/2)^2$ , where  $p_1, p_2$  are fixed points called foci.

the model does not prescribe a single loop through the lemniscate. It is very well possible for an organization to iterate the right loop a number of times before implementing a solution. For example, it may be necessary to first implement an experiment to find the cause of a problem, and then implement another experiment to find a suitable solution. Moreover, organizations might also want to implement a solution and a measurement program in parallel, to monitor the implementation of the solution.

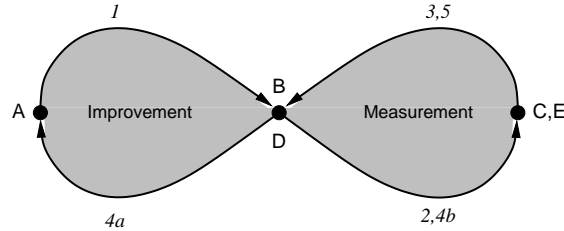


Figure 2: An example of measurement-based improvement

Let us illustrate the model by means of an example, see figure 2. Suppose a software maintenance organization has problems planning the implementation of change requests. Often, the implementation of specific change requests takes much more time than planned, and the organization fails to deliver the changed software in time. So, the problem this organization faces is the inaccurate planning of change requests (A). After analyzing the problem (1), the organization discovers that it does not know which factors influence the time needed to implement change requests (B). The organization decides to investigate this, and designs (2) a short-running measurement program (C) to investigate possible factors. After running this measurement program for a limited period of time, the gathered data are analyzed (3). We assume that a number of factors are found that influence the effort needed to implement change requests (D). Next, a planning procedure is developed and implemented (4a) in which the factors found are used to plan the change requests. An accompanying measurement program (E) is designed (4b) to gather the data needed for the new planning procedure and to monitor the accuracy of the planning (5).

We conclude this section with a few remarks on the nature of the presented generic process model of measurement-based improvement.

First, one could wonder whether this model is prescriptive or descriptive. We assume that if software organizations want to improve their processes or products, and use measurement to support those improvements, they will perform the activities as we have described above. That means we use the model as a representation – though very abstract – of what goes on in reality; i.e. it is a descriptive model. One could argue that the model is also a prescriptive model; it tells us which activities to perform when conducting measurement-based improvement. However, because of the high level of abstraction, the model is probably unsuitable to directly support organizations in their measurement-based improvement efforts.

Second, the model resembles the Goal-Question-Metric paradigm [2]. One could be tempted to map the GQM goal on the left black dot, GQM questions on the middle dot, and the GQM metrics on the right dot. However, the goal of the GQM-paradigm and the goal of the process model are not the same: the goal in the pretzel is an organizational goal, whereas the goal in the GQM-paradigm is a measurement goal. Still, GQM can very well be used to support the design of the measurement program (lower right arrow), see paragraph 3.3.

Third, the distinction made in the model between improvement on the one hand, and measurement

on the other hand, corresponds with the distinction made by Kitchenham, Pfleeger, and Fenton [9] between the empirical, real world and the formal, mathematical world. Their structural model of software measurement consists of two parts: an empirical world and a formal world. The empirical world contains entities that can have certain properties, called attributes. The formal world consists of values that measure the attributes of entities, expressed in certain units. Measurement now, is the mapping of a particular entity and attribute from the real world to a value in the formal world. The generic process model reflects the differences between these two worlds: measurement activities (the right half) are concerned with constructing a formal world based on the real world, whereas improvement activities (the left half) are concerned with changing the real world based on the formal world created by the measurement activities.

### 3 Assessing Measurement Frameworks and Guidelines

In this section we use the generic process model described in section 2 to compare different frameworks for measurement programs. In each subsection we indicate which activities and processes the respective frameworks prescribe, and position these activities and processes on the generic process model.

We discuss five different sources: the Measurement Technology Maturity Model described by Daskalantonakis, Yacobellis and Basili [5]; the Measurement Maturity Model presented by Comer and Chard [4]; the Goal-oriented Measurement Process described by Briand, Differding and Rombach [3]; the success factors for measurement programs by Hall and Fenton [8]; and the Measurement Capability Maturity Model proposed by ourselves in [10].

#### 3.1 The Software Measurement Technology Maturity Framework

Daskalantonakis, Yacobellis and Basili [5] define a framework to be used to assess the measurement technology level of an organization. The article defines five levels of measurement technology maturity, divided into 10 themes, listed in table 1.

1	Formalization of the development process
2	Formalization of the measurement process
3	Scope of measurement
4	Implementation support
5	Measurement evolution
6	Measurement support for management control
7	Project improvement
8	Product improvement
9	Process improvement
10	Predictability

Table 1: The themes of the Software Measurement Technology Maturity Framework

The five levels of maturity are similar to the Software CMM; i.e. initial, repeatable, defined, managed and optimizing. However, the model does not prescribe any processes like the Software CMM does. Instead, the model gives characterizations of each of the ten themes on each maturity level. For example, the characterizations of the third theme, ‘scope of measurement’, look as follows [5]:

- Level 1. Done occasionally on projects with experienced people, or not at all.
- Level 2. Done on projects with experienced people. Project estimation mechanisms exist. Project focus.
- Level 3. Goal/Question/Metric package development and some use. Data collection and recording. Existence of specific automated tools. Product focus.
- Level 4. Metric packages being applied and managed. Problem cause analysis. Existence of integrated automated tools. Process focus.
- Level 5. Have learned and adapted metric packages. Problem prevention. Process optimization.

The major difference between the approach taken by Daskalantonakis *et al.* and the other approaches described in this paper is that the first uses a more declarative description of the different levels of measurement technology maturity. Instead of describing the activities organizations need to implement, the results of these activities are specified. Other approaches put much more emphasis on the activities and processes themselves that organizations need to implement; i.e. they follow an imperative approach. This declarative nature of the measurement technology maturity framework makes it difficult to map it onto our generic process model. We have to translate the declarative specification of the themes into corresponding activities. Figure 3 shows our approximation of how the ten themes could be placed in the model.

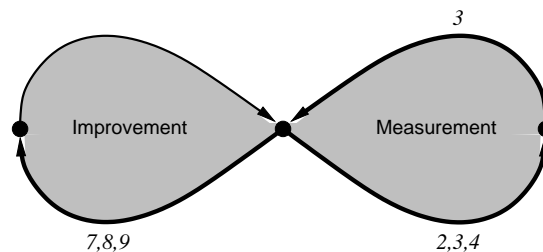


Figure 3: The Software Technology Maturity Framework mapped on the generic process model

Themes two, three, and four are concerned with the implementation of a measurement process. This measurement process includes automation of data collection, evaluation, and feedback. Themes seven, eight, and nine prescribe the usage of measurement data to improve projects, products, and processes, respectively. Unfortunately, the descriptions are more in terms of the results of the improvements than in terms of improvement activities to perform.

The other themes do not fit into the generic model. Theme one is concerned with the formalization of the development process, and essentially coincides with the Software CMM, and hence does not fit into the pretzel. We were not able to translate themes five, six, and ten into corresponding activities. Theme five, ‘measurement evolution’, is concerned with the types of measures that are taken. Measuring different kinds of measures does not necessarily change the activities needed, so we cannot place this theme in our process model. The same holds for the sixth theme, ‘measurement support for management control’. This theme is specified in terms of the type of support management receives from measurement. Different types of support do not necessarily require different activities. Theme ten, ‘predictability’, describes how the predictability of measures increases as the maturity

level of an organization increases. Again, this theme cannot directly be translated into measurement or improvement activities.

### 3.2 A Measurement Maturity Model

Comer and Chard [4] describe a process model of software measurement that can be used as a reference model for the assessment of software measurement process maturity. Unlike the maturity models described in sections 3.1 and 3.5, the measurement maturity model of Comer and Chard does not define different levels of maturity. The model consists of four key processes, derived from different sources:

- a. **Process Definition** This process includes activities such as: specification of the products, processes, and resources in need of tracking or improvement; identifying goals of the organization and the development environment; derivation of metrics which satisfy the goals.
- b. **Collection** Activities in the collection process include defining the collection mechanism, automation of the measurement gathering, implementing a measurement database, and data verification.
- c. **Analysis** Data analysis.
- d. **Exploitation** Exploitation of analyses to improve the software development process.

Unfortunately, Comer and Chard do not elaborate on the processes ‘analysis’ and ‘exploitation’, which makes it somewhat difficult to map them onto the generic process model. We assume the process ‘analysis’ consists of analyzing the gathered data, and relating the data to measurement goals. The exact borders of the ‘exploitation’ process are undefined. Especially the extent to which this process covers the actual activities needed to improve the software process remains unclear.

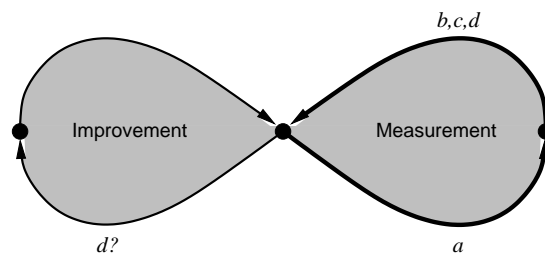


Figure 4: The Measurement Maturity Model mapped on the generic process model

Figure 4 shows how the four processes in our opinion map onto the generic process model. The process ‘exploitation’ has been placed on the lower left arrow with a question mark, because the paper provides insufficient information to decide to what extent that process is meant to cover the implementation of solutions.

### 3.3 Goal-oriented Measurement

Briand, Differding and Rombach [3] present a number of lessons learned from experiences with goal-oriented measurement. Goal-oriented measurement is described as ‘the definition of a measurement

1	Characterize the environment
2	Identify measurement goals and develop measurement plans
3	Define data collection procedures
4	Collect, analyze, and interpret data
5	Perform post-mortem analysis and interpret data
6	Package experience

Table 2: Process for goal-oriented measurement

program based on explicit and precisely defined goals that state how measurement will be used'. The process for goal-oriented measurement consists of six process steps, displayed in table 2.

During the first step the relevant characteristics of the organization and of its projects are identified. Typical questions to be posed are: What kind of product is being developed? What are the main problems encountered during projects? The characterization is intended to be mainly qualitative in nature. In the second step, measurement goals are defined, based on the characterization made during the first step. Measurement goals are defined according to Goal-Question-Metric templates [1, 2], based on five aspects: object of study, purpose, quality focus, viewpoint, and context. Having defined the measurement goals by means of the GQM templates, data collection procedures are defined during step three. Step four is concerned with the actual collection, analysis, and interpretation of the gathered data. Step five puts the data in a broader perspective by e.g. comparing the gathered data of one project with the organization baseline. The final step consists of the packaging the data analysis results, documents, and lessons learned in a reusable form.

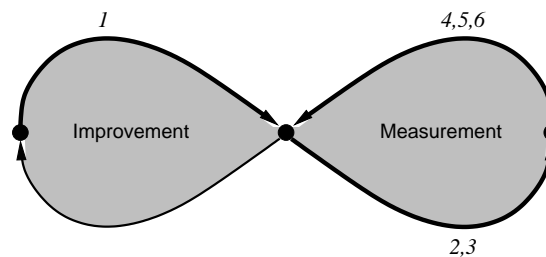


Figure 5: The goal-oriented measurement process mapped to the generic process model

Figure 5 shows how the six steps of goal-based measurement map onto the pretzel. Step one is concerned with the analysis of the organization and its problems and goals, and thus corresponds with the upper left arrow. Steps two and three deal with the translation of organizational goals into measurement goals and the design of the measurement program (lower right arrow). The last three steps consist of the collection, analysis, interpretation, and packaging of the measurement data, hence they belong to the upper right arrow.

### 3.4 Success Factors for Measurement Programs

Fenton and Hall [8] identify a number of consensus success factors for the implementation of measurement programs. Table 3 shows these factors, that were identified after studying other literature, such as [7, 11]. A closer look at the success factors shows that they are mainly targeted at reducing

1	Incremental implementation
2	Well-planned metrics framework
3	Use of existing metrics materials
4	Involvement of developers during implementation
5	Measurement process transparent to developers
6	Usefulness of metrics data
7	Feedback to developers
8	Ensure that data is seen to have integrity
9	Measurement data is used and seen to be used
10	Commitment from project managers secured
11	Use automated data collection tools
12	Constantly improving the measurement program
13	Internal metrics champions used to manage the program
14	Use of external metrics gurus
15	Provision of training for practitioners

Table 3: Consensus success factors

the risk of failure. For example, the motivation given by Hall and Fenton for factor six – usefulness of metrics data – is not that the measurement program should have added value for the organization, but rather that the usefulness should be obvious to the practitioners. From the 15 success factors, 10 are targeted at gaining the acceptance of the practitioners involved (4-9, 11, 13-15). The other five factors are concerned with reducing the risk of failure by advocating a gradual introduction and improvement of the program. The measurement program should be incrementally implemented, constantly improved, use existing materials, be supported by management, and a well-planned metrics framework should be used (1-3, 10, 12).

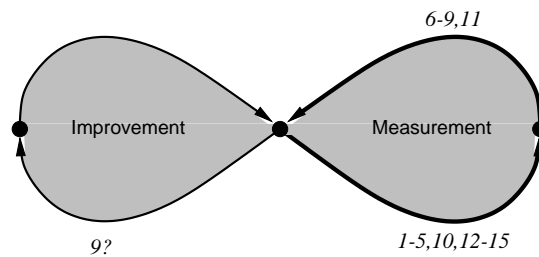


Figure 6: The success factors mapped to the generic process model

Figure 6 shows how the success factors can be mapped onto the generic process model. The majority of the success factors mentioned by Hall and Fenton refer to the implementation of measurement programs. Some are concerned with the collection and analysis part, and only one success factor is concerned with the usage of the measurement data (factor nine). That factor is marked with a question mark, because Hall and Fenton motivate it in terms of acceptance of the measurement program by the practitioners, rather than in terms of added value of the program to the company.

### 3.5 The Measurement Capability Maturity Model

In [10], we described a number of measurement program case studies. From these case studies we concluded that some organizations are better at software measurement than other organizations. Part of this difference can be explained by the fact that their *measurement capability* is higher; i.e. they are more mature with respect to software measurement. Measurement capability is defined as [10] ‘the extent to which an organization is able to take relevant measures of its products, processes and resources in a cost effective way, resulting in information needed to reach its business goals.’

Our Measurement CMM defines five different levels of organizational measurement capability, similar to the Software CMM:

1. **Initial:** The organization has no defined measurement processes, few measures are gathered, measurement that takes place is solely the result of actions of individuals.
2. **Repeatable:** Basic measurement processes are in place to establish measurement goals, specify measures and measurement protocols, collect and analyze the measures and provide feedback to software engineers and management. The necessary measurement discipline is present to consistently obtain measures.
3. **Defined:** The measurement process is documented, standardized, and integrated in the standard software process of the organization. All projects use a tailored version of the organization’s standard measurement process.
4. **Managed:** The measurement process is quantitatively understood. The costs in terms of effort and money are known. Measurement processes are efficient.
5. **Optimizing:** Measurements are constantly monitored with respect to their effectiveness and changed where necessary. Measurement goals are set in anticipation of changes in the organization or the environment of the organization.

---

2a	Measurement Design
2b	Measure Collection
2c	Measure Analysis
2d	Measurement Feedback
3a	Organization Measurement Focus
3b	Organization Measurement Design
3c	Organization Measurement Database
3d	Training Program
4a	Measurement Cost Management
4b	Technology Selection
5a	Measurement Change Management

---

Table 4: M-CMM key process areas

Each of the maturity levels is defined by a number of key process areas that an organization needs to implement. When an organization has implemented all level-two key process areas, the organization is considered to be at level two of the M-CMM. When the organization implements both the level two and three key process areas, it is at level three, etc. The key process areas of the Measurement CMM are listed in table 4, numbered by maturity level.

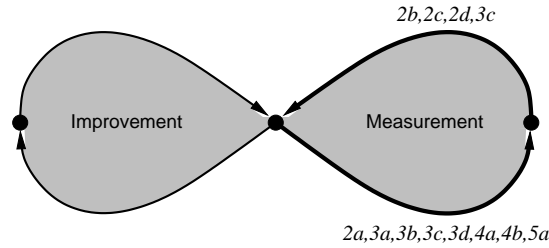


Figure 7: The M-CMM key process areas mapped to the generic process model

Figure 7 shows the M-CMM applied to the generic process model. It is not surprising that all of the key process areas map onto the right half of the ‘pretzel’. After all, we made a clear choice in the development of the Measurement CMM to focus on the measurement capability of software organizations, thereby ignoring their capability with respect to improvement. Our argument in [10] was that the improvement capability is already covered by process improvement methods, such as the Software CMM. We assumed that the organizational goals are defined outside the scope of the M-CMM, so they are invariable from a measurement point of view. The measurement process then starts with the translation of business goals into measurement goals, and ends with the interpretation of the gathered data and feedback to the owner of the business goals.

#### 4 Differences and Similarities

In the previous section, we have compared five different measurement program frameworks, using the generic process model for measurement-based improvement. There are a number of issues that deserve attention.

First, if we look at the intersection of the guidelines provided by the different approaches, we see that there is quite some consensus on what activities are needed to successfully design and implement measurement programs. Though the frameworks all stress different aspects of measurement programs, they agree on the basics of measurement programs, such as practitioner support, proper data analysis, feedback, etc.

Second, each of the approaches seems to offer guidelines that the other approaches do not offer. This probably is partly due to the different structure and nature of the frameworks. However, it does suggest that beyond the basic requirements for measurement programs, there are a number of issues on which consensus has not been reached yet. For example, only Briand *et al.* prescribes the packaging of measurement experiences in a reusable form. Of the five frameworks, Hall and Fenton are the only ones to advocate the use of external measurement guru’s.

Third, if we look at each of the pretzels used in section 3 to show how the activities of the different approaches map onto the measurement-based improvement process, we see that none of the five frameworks covers the complete cycle. Either the frameworks only cover measurement activities, or they only partly cover the improvement activities. One could argue that this is not illogical, since these measurement program frameworks focus on the implementation of measurement programs, and not on improvement activities. However, failure factors for measurement programs suggest otherwise. Take for example the failure factors for measurement programs as suggested by Verdugo, reported in [6, p. 511]:

1. Management does not clearly define the purpose of the measurement program and later sees the program as irrelevant.
2. Systems professionals resist the program, perceiving it as a negative commentary on their performance.
3. Already burdened project staff are taxed by extensive data-collection requirements and cumbersome procedures.
4. Program reports fail to generate management action.
5. Management withdraws support for the program, perceiving it to be mired in problems and 'no-win' situations.

From these failure factors we see that both support from practitioners for the measurement program, as well as management support, is important. A measurement program will fail if practitioners do not see the value of it, but it will also fail if management fails to take action based on the generated data. This means that a successful measurement program needs more than carefully designed metrics, accurate data analysis, measurement databases, etc. It needs to be used. Any measurement program that does not generate any action is to be considered a failed measurement program.

## 5 Conclusions

In this paper we have introduced a generic process model for measurement-based improvement. We have used this model as a reference model to assess five different measurement program frameworks. These frameworks have in common that they all provide guidelines on how to implement and improve measurement programs. From the assessments we conclude that:

- there is quite some consensus on the basic activities needed to successfully implement measurement programs; but,
- at the same time, different frameworks emphasize widely different aspects of measurement program implementation.

In addition, the assessment also reveals that:

- there is almost no consensus on, nor description of, activities needed to successfully *use* the results from measurement programs.

In a way, this is quite surprising. Usage of the results of measurement programs is probably the most important indicator of success: if the results are not used, the measurement program is doomed to fail. Still, the approaches to implementing and improving measurement programs described in this paper put little emphasis on the actual usage of the measurement results. They mostly concentrate on the implementation of the measurement programs themselves. They provide guidance on the derivation of metrics from goals, suggest a staged implementation, prescribe measurement protocols, etc. But, they do not tell us how to ensure that action will be taken, based on the measurement data.

We do not claim that this is necessarily a shortcoming of the presented frameworks. Most of them implicitly or explicitly focus on the activities needed to successfully implement measurement programs. Nevertheless, organizations implementing measurement programs need to be aware that the usage of measurement program outcomes to improve processes or products is at least as important as a 'correct' implementation of the program. After all, like the proof of the pudding, or pretzel, is in the eating, the proof of software measurement is in its usage for improvement.

## Acknowledgments

This research was partly supported by the Dutch Ministry of Economic Affairs, projects ‘Concrete Kit’, nr. ITU94045, and ‘KWINTES’, nr. ITU96024. Partners in these projects are Cap Gemini, Twijnstra Gudde, the Tax and Customs Computer and Software Centre of the Dutch Tax and Customs Administration, and the Technical Universities of Delft and Eindhoven.

## References

- [1] Victor Basili, Lionel Briand, Steven Condon, Yong-Mi Kim, Walcélío L. Melo, and Jon D. Valett. Understanding and Predicting the Process of Software Maintenance Releases. In *Proceedings of the 18th International Conference on Software Engineering*, pages 464–474, Berlin, Germany, May 25-29, 1996. IEEE Computer Society Press.
- [2] Victor R. Basili and H. Dieter Rombach. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.
- [3] Lionel C. Briand, Christiane M. Differding, and H. Dieter Rombach. Practical Guidelines for Measurement-Based Process Improvement. *Software Process – Improvement and Practice*, 2(4):253–280, December 1996.
- [4] Peter Comer and Jonathan Chard. A measurement maturity model. *Software Quality Journal*, 2(4):277–289, December 1993.
- [5] Michael K. Daskalantonakis, Robert H. Yacobellis, and Victor R. Basili. A Method for Assessing Software Measurement Technology. *Quality Engineering*, 3:27–40, 1990-1991.
- [6] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. Int. Thomson Computer Press, second edition, 1997.
- [7] Robert B. Grady. *Practical software metrics for project management and process improvement*. Hewlett-Packard Professional Books. Prentice-Hall, Inc., 1992.
- [8] Tracy Hall and Norman Fenton. Implementing Effective Software Metrics Programs. *IEEE Software*, 14(2):55–65, March/April 1997.
- [9] Barbara Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. Towards a Framework for Software Measurement Validation. *IEEE Transactions on Software Engineering*, 21(12), December 1995.
- [10] Frank Niessink and Hans van Vliet. Towards Mature Measurement Programs. In Paolo Nesi and Franz Lehner, editors, *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering*, pages 82–88, Florence, Italy, March 8-11, 1998. IEEE Computer Society.
- [11] Shari Lawrence Pfleeger. Lessons Learned in Building a Corporate Metrics Program. *IEEE Software*, 10(3):67–74, May 1993.