Towards Mature Measurement Programs*

Frank Niessink and Hans van Vliet
Faculty of Mathematics and Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 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 organizations are using measurement as a means to improve their software development and maintenance processes. A reasonable consensus has been reached about the main success factors for measurement programs. However, no comprehensive approach has so far been published for the processes that need to be in place to ensure effective and efficient measurement. We propose a capability maturity model for measurement that can be used to both assess the measurement capability of software organizations and to identify directions for improvement of their measurement capability.

This 'Measurement-CMM' originates from our efforts to establish measurement programs in a variety of settings. These efforts had mixed results, and our analysis thereof showed widely different measurement capabilities amongst the organizations involved. A measurement maturity scale similar to that of the Software-CMM allowed us to explain many of the differences observed. At the same time, it suggests ways to improve on this.

1. Introduction

One of the first major publications on the topic of measurement programs was the 1987 book by Grady and Caswell [6]. Since then, many reports have been published on the use of software measurement to measure complexity, reduce faults, improve quality and improve processes. According to Fenton and Pfleeger: 'Software measurement, once an obscure and esoteric specialty, has become essential to good software engineering.' [5].

In this paper we want to discuss the organizational side of software measurement. More specifically, we want to address the following questions:

- How to introduce measurement in a software organization? What are the necessary steps to set up a measurement program and in which order should they be performed? How can existing measurement programs be enhanced?
- What are the prerequisites that need to be satisfied before we can set up a measurement program or take certain steps to improve the program?
- What is – or what should be – the relation between measurement and the maturity of the software process?

In the past two years we have established measurement programs in a variety of settings. These efforts had mixed results. An analysis thereof showed that the participating organizations had widely different measurement capabilities. A measurement maturity model akin to the Capability Maturity Model[1], (CMM) [12] allowed us to explain many of the differences observed, and provided us with initial answers to the above questions.

Four case studies of software measurement are described in the next section. In section 3, we look at the literature pertaining to the successful introduction of measurement programs in organizations. Section 4 describes our proposal for a measurement capability maturity model as a means of structuring the introduction and enhancement of software measurement in organizations. Finally, section 5 presents our conclusions and gives indications for further research.

2. Case studies

Since 1995 we have been involved in a project aimed at improving the quality of IT services, such as IT infras-

*Proceedings of the Euromicro Working Conference on Software Maintenance and Reengineering, pp. 82-88, Florance, Italy, March 8-11, 1998.
[1]Capability Maturity Model and CMM are service marks of Carnegie Mellon University.

tructure management, software maintenance and exploitation of information systems. The research focuses on improving the state-of-the-art with respect to the management of Service Level Agreements (SLAs) between IT service providers and their clients [17]. Service Level Agreements include performance indicators of various kinds that together specify the service that is to be delivered. Measurement plays an important role in this context:

1. Measurement of the service levels is needed to report the quality of the delivered service to the customer.
2. The service provider needs measurement to gain insight in his own abilities to be able to draw up realistic SLAs.
3. The service provider needs measurement to identify possible areas for improvement and track the improvement process.

To investigate these roles of measurement, we established measurement programs in various software maintenance settings, with mixed results. The environments in which we implemented a measurement program can be characterized as follows:

A. This organization is a business unit of a major Dutch software house. It maintains and operates information systems for several small and large customers. In 1994, a questionnaire was developed to gather project information in order to support the bidding process for new projects and new Service Level Agreements.

   Estimates were to be derived using an experimental tool developed in the course of an ESPRIT project. There was very little knowledge of the tool and the underlying statistics within the business unit. The unit had undergone several reorganizations in the last few years, and was still very much in a state of flux. The accuracy of the data gathered was, at best, mixed. Quite a few entries were left blank. Quite a few also suspiciously looked like guesstimates. The questions posed sometimes asked for a subjective answer, and were sometimes ambiguous.

B. Organization B maintains and supports a large financial administrative information system for one of the ministries of the Dutch government. Here, maintenance function points – a variant of the standard Albrecht function points – are used to support negotiations between the users and the maintainers about changes to the system [11].

   This seemed like a measurement-wise organization. There were detailed guidelines as to how and what to measure. The measurement process was strongly supported by management. Everybody knew what the measurements were used for. There were clearly visible piecharts illustrating various performance indicators on the door of the manager's office.

C. Organization C is the IT department of a large organization, responsible for carrying out the Dutch social security system. The measurements were done at three of the so-called product teams, which are teams of about 15 engineers, each team being responsible for the maintenance of a number of information systems. The goal of the measurement program was twofold: to gain insight into the cost drivers for change requests and to gain practical experience with the introduction of a measurement program.

   This is also an organization in flux. It is part of a large organization that has been split up and gone commercial. The people still have to get accustomed to their new role: from being an internal maintenance department to being a commercial service provider. The setting up of a measurement program as well as collecting data and analyzing them were done by an MSc student as part of his graduation project. Participants were willing to help, but their attention easily slipped. Management's goals were primarily aimed at establishing a sound organization, and taking measurements was supported insofar this helped to reach the primary goals.

D. The final organization is the IT department of a large Dutch industrial organization. The measurements took place at two departments, responsible for the maintenance of several administrative systems and process-control systems. The goal of this measurement program was identical to the goal of organization C. As in organization C, the setting up of the measurement program as well as the collection and analysis of data was done by a (different) MSc student as part of a graduation project.

   This organization is a stable organization. Its primary process concerns the production of steel. For each information system, there is an intermediary between the client and the maintenance department. This intermediary is located at the client site. He is responsible for the phrasing of change requests. He is in direct contact with the programmer(s) in the maintenance department. The amount of analysis and design done at the client side varies per system. Budgets are allocated per system per year. There is some pressure from the client side to make maintenance costs more "visible". The measurement program was started because of this pressure.

The measurement program at organization A can be denoted as a failure, whereas the measurement program at organization B is a success. The measurement programs in organizations C and D are successful to a certain extent, but are not yet solidly embedded in the organizations. The ques-

Incremental implementation
Well-planned metrics framework
Use of existing metrics materials
Involvement of developers during implementation
Measurement process transparent to developers
Usefulness of metrics data
Feedback to developers
Ensure that data is seen to have integrity
Measurement data is used and seen to be used
Commitment from project managers secured
Use automated data collection tools
Constantly improving the measurement program
Internal metrics champions used to manage the program
Use of external metrics gurus
Provision of training for practitioners

Table 1. Success factors (taken from [8])

tion is: what causes these differences, and how can they be overcome?

3. Enhancing Measurement Capability

Several authors have identified success factors for measurement programs, e.g. [8, 9, 16]. After studying other research on measurement, Hall and Fenton [8] identified a consensus on requirements for measurement program success. Table 1 displays those success factors.

However, these success factors do not provide organizations with a clear cut path on how to introduce measurement into their organization, i.e. which steps need to be taken first and which processes need to be in place. It seems a logical step to try to develop a measurement improvement method based on these consensus success factors. Before we go deeper into this, we first take a look at the role measurement plays in software process improvement methods.

With respect to software process improvement, different approaches exist, most notably the Software Capability Maturity Model (S-CMM) [2, 12, 13]. Other improvement paradigms for software are largely based on the S-CMM, including the BOOTSTRAP approach [7, 10] and the initiative to develop a suite of standards for software process assessment and improvement (SPICE) [4]. Each of these improvement models includes measurement as a means to help improving the software process. However, these methods do not prescribe how the measurement processes themselves should be implemented. For example, the S-CMM does prescribe that in all key process areas measurements should be taken to determine the status of the activities. But only on level 4, measurement is explicitly dealt with by the key process area Quantitative Process Management. Since the S-CMM is concerned with the software process, measurement is only covered insofar it directly deals with improving the software process. The issue of introducing and improving measurement processes is beyond the scope of the S-CMM proper.

On the subject of the relation between measurement and software process improvement, Pfleeger and McGowan [15] recommend the collection of different measures depending on the organization's maturity level. See table 2. In [14], Pfleeger presents a combinational approach to measurement programs, using the Goal-Question-Metric paradigm [1] to derive goals to be met and questions to be answered, and the S-CMM to decide what can be measured – i.e. what is visible. As the process matures, visibility increases and a more comprehensive set of metrics can be measured.

None of these sources gives a structured path to enhance *measurement capability*. The success factors for software measurements, though highly useful, do not differ all that much from the hit list for software reuse, formal specifications, or any major organizational change relating to the software process. They give premises for success, not roads to get there. In a similar way, Pfleeger's work [14, 15] gives insight into which measures can be collected at which maturity level. It does not help us to improve the measurement process itself. Our Measurement-CMM is intended to fill that gap.

4. The M-CMM

In this section we describe the proposed Measurement Capability Maturity Model. First, the objectives of the M-CMM are laid out. Next, the maturity levels of the M-CMM are described. Section 4.3 touches on the key process areas and finally section 4.4 describes the relation between the Measurement-CMM and other maturity models.

Level	Measures
5. Optimizing: improvement fed back to process	process and feedback for changing process
4. Managed: measured process (quantitative)	process and feedback for control
3. Defined: process defined, institutionalized	product
2. Repeatable: process dependent on individual	project
1. Initial: ad hoc	baseline

Table 2. Process maturity related to measures (adapted from [15])

4.1. Primary objectives of the M-CMM

The goal of the M-CMM is twofold:

1. to enable organizations to assess their cababilities with respect to both software and software process measurement, and,
2. to provide organizations with directions and steps for further improvement of their measurement capability.

The M-CMM does this by measuring the measurement capability maturity on a five level ordinal scale and by prescribing processes that have to be in place in order for an organization to reside on that level. This is roughly the same framework as used in the Software-CMM [2], or the People-CMM [3].

We define *measurement capability* as '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.'

An organization that scores high on the M-CMM scale will be able to:

- gather relevant information about its own performance with respect to its long and short term business goals;
- continue to collect the relevant information when either the organization itself or its environment changes;
- do so in a cost effective way by reducing the number of collected measures or by using automated measure collection when possible;
- provide an environment in which both management and staff are convinced of the usefulness of measurement and, moreover, are continuously being convinced by the measures themselves.

Note that the business goals themselves are not part of the model, they are input to the model. Measurement goals are derived from the business goals. Organizations with a higher measurement capability are better able to measure the right measures in order to help reach their business goals.

Also note that measurement capability addresses the ability of organizations to measure processes, products and resources *as is*. Improvement of the software process or products is not part of the M-CMM, though higher visibility (i.e. maturity) offers more opportunities to measure, see [15]. For example, let us suppose that an organization wants to know how much time it spends on testing, but the organization does not follow a defined development cycle in which it is clear when the testing phase starts and when it ends. In such a case the organization cannot expect to take valid measurements of time spent on testing without clearly specifying what is meant by testing and without making sure everyone is working according to that specification. Similarly, implementing a configuration management system to ensure that software components are uniquely identifiable is not part of the M-CMM. While these software process improvements do improve visibility of the software process, they do not improve *measurement capability*. Moreover, they are already part of software process improvements methods, such as the Software-CMM.

4.2. The maturity levels of the M-CMM

The maturity levels for the Measurement-CMM are defined similarly to those of the other capability maturity models. This means that on level 1 – initial – there are no key process areas defined. In essence, level 1 is the level on which all organizations reside that have no key process areas implemented. On level 2 – the repeatable level – organizations have basic measurement processes in place, which means they are able to collect measures during projects. Measures are probably not comparable across projects, since each project potentially has its own measurement goals and defines its own measures. On level 3 – the defined level – this problem is solved, because the organization standardizes its measurement process and determines a basic set of measures that each project has to collect. Also, an organization wide measurement database

is created, which contains all historic project data. Level 4 is the managed level, meaning that the organization will be able to assess the costs of different measures. Technology is being used to make the measurement process more efficient. Finally, at level 5 – the optimizing level – the organization is ensuring that measurement processes are not only efficient, but also effective. Measures are regularly judged on their merits and measurement processes are adjusted when necessary to reflect changes in the measurement environment.

More formally, we define the M-CMM maturity levels as follows:

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 analyse 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.

4.3. The key process areas of the M-CMM

For an organization to reach a certain level other than the first level, certain processes need to be in place. These processes are grouped in key process areas, where *key* merely means that there could be more – non-key – processes, but that those non-key processes do not need to be in place to reach a certain maturity level. An organization can only reach a certain maturity level when it has implemented all key process areas for that level.

Below we present the key process areas for the M-CMM. Note that each of these key process areas should be described more thoroughly, in terms of goals and common features (common features define the activities performed and the activities needed to institutionalize the process, see [2]). For reasons of space we only specify the purpose of each key process area:

1. Initial: no key process areas.

2. Repeatable:

   (a) Measurement Design: Measurement goals, measures and measurement protocols are established according to a documented procedure, and goals, measures and protocols are kept consistent with each other. Measurement protocols are managed and controlled.

   (b) Measure Collection: Measures are collected according to the measurement protocol.

   (c) Measure Analysis: The collected measures are analyzed with respect to the measurement goals.

   (d) Measurement Feedback: The measurement goals, the measurement protocols, the collected measures and the results of the analysis are made available to the people involved in the measurement process.

3. Defined:

   (a) Organization Measurement Focus: Software measurement activities are coordinated across the organization. Strengths and weaknesses of the measurement process are identified and related to the standard measurement process.

   (b) Organization Measurement Design: A standard measurement process for the organization is developed and maintained and information with respect to the use of the standard measurement process is collected, reviewed and made available.

   (c) Organization Measure Database: Collected measures are stored in an organization-wide database and made available.

   (d) Training Program: People are provided with the skills and knowledge needed to perform their roles.

4. Managed:

   (a) Measurement Cost Management: The costs of measurement are known and used to guide the Measurement Design Process and the Organization Measurement Design process.

   (b) Technology Selection: The information of measurement costs is used to choose and evaluate technology support for the measurement process.

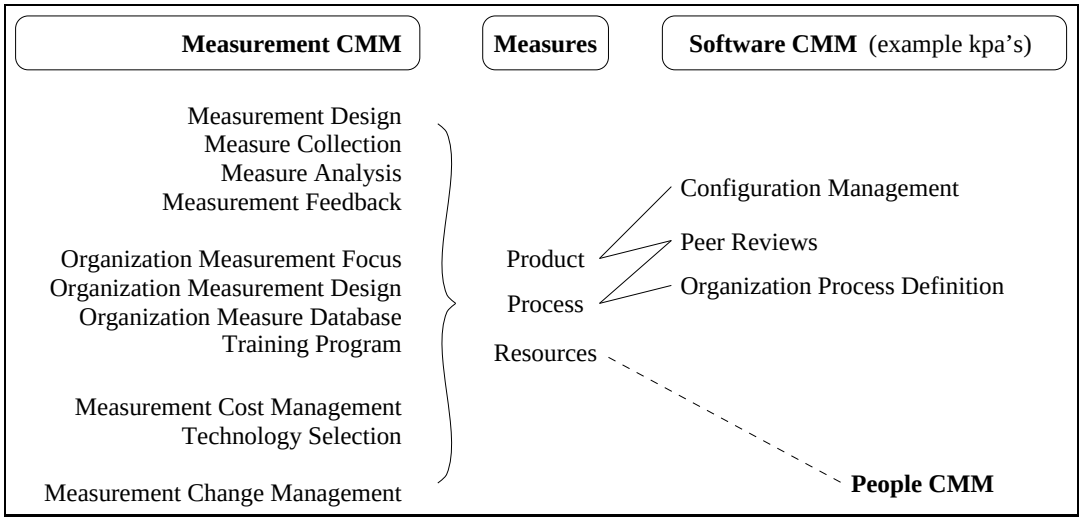

Figure 1. The M-CMM linked with other CMMs

5. Optimizing:

   (a) Measurement Change Management: The measurement capability is constantly being improved by monitoring the measurement processes and by anticipating changes in the software process or its environment.

The M-CMM maturity levels together with the key process areas provide organizations with both a measurement scale along which they can assess their measurement capability, and directions for future improvements.

4.4. M-CMM and other capability maturity models

As mentioned in section 4.1, the Measurement-CMM does not prescribe the improvement of processes other than measurement processes. The improvement of software processes is covered by the Software-CMM. The two models are linked by the processes, products and resources that are subject to measurement on the one hand, and are part of the software process – and thus covered by the Software-CMM – on the other hand. The same goes for the relationship between the M-CMM and the People-CMM. We can visualize this as in figure 1.

We can see that an organization that has reached level 2 of the Software-CMM is able to take detailed measures about software components that are under configuration management. On the other hand, if the organization does not have a standard software process, it will be difficult – if not impossible – to measure the duration of software activities, since they have not been standardized. See [14, 15].

We observe that an organization that wants to increase the knowledge about and insights into its own software processes, needs to advance on both the S-CMM and the M-CMM ladder.

5. Conclusions and future research

If we apply the Measurement Capability Maturity Model to the environments discussed in section 2, we observe the following:

- Organization A is at level 1. None of the key process areas of level 2 has been fully implemented. The attempt to build an organization-wide project database clearly was a bridge too far. Our current attempts concentrate on improving the measurement design and collection process areas.

- Organization B is at level 2. Since the organization is concerned with one project only, one is tempted to conclude that it is at level 3 as well. However, current measurements are used for one goal only, viz. estimate the size of changes. When we tried to relate these size data to other resource and product characteristics, we encountered considerable difficulties. Clearly, process areas such as Organization Measurement Focus and Organisation Measure Database are not fully developed yet. Our next step is to implement these.

- Both organization C and organization D are at level 1. The MSc projects concerned all of the process areas of level 2. Clearly, none of these processes are firmly embedded within the organizations yet. For that reason, the measurement program is still fragile.

Current efforts are directed at embedding the level 2 key processes in the organizations.

The Measurement Capability Maturity Model provides us with the instruments to assess the various environments in which we implemented a measurement program. It allows us to assign a measurement score to each of the organizations, and explains the success or failure of our measurement efforts. It also identifies areas in which improvements can be sought.

Evidently, the M-CMM needs to be further validated. It must be refined and tuned to make it a useful assessment tool. As noted in section 2, our research is concerned with improving the state-of-the-art with respect to the management of Service Level Agreements (SLAs) between IT service providers and their clients. If we are able to determine the measurement capability of the service provider, we are able to determine the level of detail and accuracy to be aimed at when drawing up an SLA. Such will result in more realistic SLAs on the short term. In the long run, we will be able to improve the Service Level Management process, and turn the IT service provider organization into a learning organization.

6. Acknowledgements

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. We would like to thank the MSc students Kit Lam and Jack Hulscher who implemented the measurement programs at organizations C and D.

References

[1] V. R. Basili and H. D. Rombach. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.

[2] Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process.* SEI Series in Software Engineering. Addison-Wesley Publishing Company, 1995.

[3] B. Curtis, W. E. Hefley, and S. Miller. Overview of the People Capability Maturity Model. Technical Report CMU/SEI-95-MM-01, Software Engineering Institute/Carnegie Mellon University, Sept. 1995.

[4] K. El Emam, J. Drouin, and W. Melo, editors. *SPICE: The theory and practice of software process improvement and capability determination.* IEEE Computer Society Press, 1997.

[5] N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* Int. Thomson Computer Press, second edition, 1997.

[6] R. B. Grady and D. L. Caswell. *Software Metrics: Establishing a company-wide program.* Prentice-Hall, Inc., Hewlett-Packard Company, 1987.

[7] V. Haase, R. Messnarz, G. Koch, H. J. Kugler, and P. Decrinis. Bootstrap: Fine-Tuning Process Assessment. *IEEE Software*, 11(4):25–35, July 1994.

[8] T. Hall and N. Fenton. Implementing Effective Software Metrics Programs. *IEEE Software*, 14(2):55–65, March/April 1997.

[9] R. Jeffery and M. Berry. A Framework for Evaluation and Prediction of Metrics Program Succes. In *Proceedings of the First International Software Metrics Symposium*, pages 28–39. IEEE Computer Society TCSE, IEEE Computer Society Press, 1993.

[10] P. Kuvaja, J. Similä, L. Krzanik, A. Bicego, G. Koch, and S. Saukkonen. *Software Process Assessment and Improvement: The BOOTSTRAP Approach*. Blackwell Publishers, 1994.

[11] F. Niessink and H. van Vliet. Predicting Maintenance Effort with Function Points. In M. J. Harrold and G. Visaggio, editors, *Proceedings of the International Conference on Software Maintenance*, pages 32–39, Bari, Italy, October 1-3, 1997. IEEE Computer Society.

[12] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber. Capability Maturity Model for Software, Version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute/Carnegie Mellon University, Feb. 1993.

[13] M. C. Paulk, C. V. Weber, S. Garcia, M. B. Chrissis, and M. Bush. Key Practices of the Capability Maturity Model, Version 1.1. Technical Report CMU/SEI-93-TR-025, Software Engineering Institute/Carnegie Mellon University, Feb. 1993.

[14] S. L. Pfleeger. Maturity, Models and Goals: How to Build a Metrics Plan. *The Journal of Systems and Software*, 31(2):143–155, Nov. 1995.

[15] S. L. Pfleeger and C. McGowan. Software Metrics in the Process Maturity Framework. *The Journal of Systems and Software*, 12(3):255–261, July 1990.

[16] S. Rifkin and C. Cox. Measurement in Practice. Technical Report SEI-91-TR-16, Software Engineering Institute/Carnegie Mellon University, July 1991.

[17] J. Trienekens, M. van der Zwan, F. Niessink, and J. van Vliet. *De SLA Specificatiemethode.* Cap Gemini Perform Service Management. Academic Service, 1997. (In Dutch).