Key Developments in the Field of Software Productivity Measurement

Gibeon Soares de Aquino Júnior[1,2] and Silvio Romero de Lemos Meira[1,2]

[1] C.E.S.A.R. - Recife Center for Advanced Studies and Systems, Recife, Brazil
[2] Federal University of Pernambuco, Recife, Brazil

Abstract. The search for productivity improvement has been a growing concern of organizations in recent time. To achieve this goal is necessary, first, to know how to measure the productivity. The wide literature about this subject confirms the interest about this topic. Several questions already reside and there are many myths about the productivity measurement. This paper conduct an investigation about the productivity metrics and measurement studies published in literature, categorize the productivity metrics indentified according to the type of measure used and identify the main lessons learned about this matter. Based on the literature review and on analyses of proposed metrics found, we propose some directions in order to evolve the state of the art on software productivity measurement and, consequently, on the software productivity theme.

1 Introduction

The global environment requires that companies increasingly improve their levels of quality and reduce their production costs. The main way to reduce production costs, specifically in software development, is through the increasing of productivity, in other words, develop further with a lower cost or effort. For this reason, the search for improving productivity has been a constant concern in organizations. However, to improve productivity it is necessary, first, to understand how to measure it. DeMarco [1] emphasizes that the first step towards improve and control is to measure, as his statement says: "*You can't control what you can't measure*". Therefore, this article in particular, will explore the question: **How to measure productivity?**

Productivity is conceptually a simple measure , and in general can be defined as the ratio between what is produced and what is consumed: $productivity = output/input$. The concept of productivity is widely used in many areas of expertise, from agriculture to economics. Despite the idea of productivity being very simple and popular in other areas, specifically in software development it is a historically complex problem. According to Jones [2] the lack of precise and non-ambiguous definition of productivity metric has been a source of problems in software projects. This difficulty in measuring the productivity in software development is not caused by the productivity itself, but by the intrinsic characteristic that is the meaning of *output* in software projects context. Quantify the results produced in a software project in terms of size, complexity or value to the customer is a very complex problem [3].

However, Scacchi [4], reports on its study of productivity metrics that it is evident that the studies on this matter is fundamentally inadequate and potentially flawed. Moreover, depending on how and which indicators are measured the conclusions about the productivity can be completely different.

The extensive literature on productivity and productivity measurement confirms the scientific interest on this subject. Many issues are still without answers and there are still many myths about the problem of productivity. This article investigates several studies published in literature on metrics and measurement of productivity in software development projects in order to raise the main lessons learned on this matter. In particular, the Section 2 describes the first discussions and studies published in the literature about the productivity improvement and measurement. The Section 3 describes the studies about productivity measurement (a literature review) and categorizes them according to the type of measurement used (Sections 3.1, 3.2, 3.3 and 3.4). Moreover, we indentified some trends in the literature review (Section 3.5), analyzed some related works (Section 4) and proposed some future directions in software productivity measurement researches to improve the state of art in this matter (Section 5).

2 The Genesis

Despite the emphasis on productivity in nowadays, this problem is not new. At the beginning of Software Engineering in 1968, NATO Conference, where about 50 experts in computing, 11 countries gathered in Garmisch, Germany, to discuss the problems of software development at the time [5], the concern with productivity was already mentioned. Even during the discussions were cited several factors affecting productivity, such as programming language, programmer experience, culture and motivation, which are still well supported by studies in the area.

Additionally, previous studies have demonstrated the importance of the problem. The first empirical studies, based on statistical data analysis from the U.S. Army software development projects, appeared in the mid-60 in the SDC (System Development Corporation) [6, 7]. They had as goal the definition of factors affecting the cost of software projects and, as result, improving the process of estimation. Although these works have not had the direct purpose of productivity analysis, they can be considered part of this scope since identified several factors affecting productivity besides measuring the projects productivity.

Later, several studies on the same area in the SDC, have extended the previous work exploraring the problem with data from new projects and based on the results of the practical application of previous results [8–10]. In particular, LaBolle [11] published a cost estimation model, resulting of an extensive study done by the investigation of 169 finalized projects. This study have statistically analyzed various factors that had influence on the final development effort, or more specifically, on the software projects productivity. Other more specific studies on the problem have emerged over time too [12, 13]. Most of them were part of a great search for better estimation methods, which allowed a better reliability in predicting the costs of software projects and thus avoiding the typical problem of this area, which became known as the Software Crisis [5].

Throughout the Software Engineering history, many studies have been conducted related to the problem of productivity and as could be seen since 60's this matter was already discussed. Although there are several aspects of the problem still open, there are, on the other hand, a large range of relevant results which demonstrate the possibilities of exploitation of this issue. Therefore, this area of research is very promising and relevant from the scientific viewpoint.

3 Productivity Measurement: A Survey

Measuring productivity of projects has been a major challenge for practitioners and researchers in software development. Despite the difficulty of effectively measuring the software projects productivity, much effort has been performed in attempt to obtain ways to assess productivity, following the philosophy of Gilbs's law [14] who state that *"Anything you need to quantify can be measured in some way that is superior to not measuring it at all"*.

Due to the large numbers of studies, using different metrics of productivity, it is essential to group them with the aim of effectively investigate them. The studies reviewed in this section shall be grouped according to the approach of *output* measurement and will be described in more details in the following sections.

3.1 Physical Size Based Metrics

This section considers the studies that used SLOC, variations or derivations thereof. Examples of these SLOC variations are *Delivery Source Instructions*, *Executable Lines Of Code* and *Developed Statements*. The derivations of SLOC metrics are calculated from the source code, such as *Non-Commented Source Code* and *Effective Source Lines of Code*.

Although SLOC being something quite basic it is widely used in studies of productivity as the output measure in the evaluation of productivity. In studies that follow we identify that different metrics are used for output measurement, but all based on the counting physical lines of code. Different names are used to represent the same concept, such as *Delivery Source Lines*, *Machine Instructions*, *Delivery Source Instructions* and *Executable Lines Of Code*. Another difficulty encountered when compiling these studies was that on most of these, there is no precise specification of what is considered in the counting of these metrics (e.g., comments, blank lines, statements, etc). And because this, further analysis and comparison between them becomes more difficult.

Initial studies about this matter [15, 11, 9] used a metric conceptually equal to SLOC: *Man-Months per Machine Instructions*. Aron [13], in his studies on projects at IBM, widely used a metric of SLOC. Subsequently, Walston and Felix [16] conducted a broader study in the IBM Federal Systems Division and used DSL/MM as the *productivity* metric, where DSL (Delivery Line Source) means only those lines of code delivered as part of the product.

At the end of the '70s, Jones [2] states that productivity and quality metrics, in terms of lines of code are paradoxical, since LOC per unit of effort tends to emphasize more size and format of the programs than the efficiency and quality of these. According to him, this metric tends to penalize high-level languages for languages such as Assembly. For this reason the LOC is not a good indicator of economic productivity, despite being frequently used. Boehm [3], for example, showed that a well formatted program can be three times its original size.

Even recognizing the shortcomings of SLOC based metrics, in the 80s the scientific community, as was the case of Lawrence [17], Bailey and Basili [18], Boehm [19, 20] and Vosburgh et al. [21] continued to use it in their studies of productivity and estimation. Grady and Caswell [22] and Card et al. [23] used the metric of $NCSS$, based on SLOC, to evaluate the productivity of projects in HP and Software Engineering Laboratory (SEL), respectively. Recently, studies on productivity also used SLOC metric, such as MacCormack [24], Refer [25] and Moses et al. [26].

3.2 Functional Size Based Metrics

This section considers the studies which use *output* measures based on number of features perceived by the user. The most common examples of these types of metrics that are **FP** (Function Points) [27] and **UCP** (Use-case Points).

Behrens [28] widely used metrics based on FP to measure the projects productivity. Banker [29], Parkan et al. [30], Jones [31] and Moses et al. [26] also used FP as *output* measure. On the other hand Symons [32] and Maxwell and Forselius [33] criticizes the use of FP and suggest adjustments to the metric.

About UCP, it is not so popular as FP, but some studies reports a good use of it. Arnold e Pedross [34], for example, used it as measure of *output* in their study. We encountered several studies reporting it uses only for estimation purpose [35, 36] and not for productivity assessment. Moreover, because it is not so popular, there are some proposals which adapts FP measurement to be used in environments where use cases, class diagrams and sequence diagrams are used [37, 38].

FP became one of the most known and used metric to quantify the size of software and consequently the productivity for information systems and real time systems. Nowadays it is largely used in industry, with support of world wide associations which maintains and evolve it, as IFPUG[3] and NESMA[4].

Although this, it is very criticized and questioned because several assumptions and definitions do not have a theoretical apparatus that validate it [39, 40, 31].

3.3 Design Based Metrics

At this group, studies that use metrics related to the design are present, they use metrics as number of modules, classes or related things. These metrics have a correlation with the SLOC, but are based on different aspects.

[3] IFPUG: International Function Point Users Group – http://www.ifpug.org/

[4] Netherlands Function Point Users Group – http://www.nesma.nl/

Chatman [41], for example, suggests using a new metric, called *Change-Point*. Moser and Nierstrasz [42] used a new metric called System Meter. The System Meter is an object oriented metric and is calculated as the sum of the sizes of all objects in the system. Morasca and Russo [43] used three different metrics to assess the size of productivity, they were: Function Points, LOC and Number of Modules. The conclusion of the authors was that the first is useful for the outside perspective (Customer) and the latter two for the internal (Team and Organization).

3.4 Value Based Metrics

Studies at this group use a more modern vision to assess the productivity and use *output* metrics based in added value, or use multidimensional models that assess different aspects of what is produced in a software project.

Duncan [44] criticizes the use of SLOC based metrics as a productivity measure. He says the key process of software development is the transformation of ideas into products. So to measure the real productivity of software development, we need to measure how effectively and efficiently it can transform ideas into software. For Anselmo and Ledgard [45], measure the amount of features of software, using traditional techniques such as Function Point and Use case point is insufficient, since such methods assess so simple the level of functionality complexity.

According to Hastings and Sajeev [46] the metric of function point does not adequately address the internal complexity of the system, which can result in a disproportionate measure for each type of software. Based on this they proposed a new measure, known as *Vector Size Measure (VSM)*, which considers both the complexity and functionality to calculate the size of software, i.e. the unit of output.

Stensrud and Myrtveit [47] criticize the traditional productivity measurement model based on the equation $productivity = output/input$, because according to them a project of software has various measures *output* and therefore a multidimensional model would be more appropriate to effectively evaluate the project productivity. In this work they propose to use a metric called *Data Envelopment Analysis Variable Returns to Scale (DEA VRS)*, which considers multiple variables of *input* and *output*.

Following the same idea of using a multidimensional approach to assess productivity, Abran and Bunglione [48] define a three-dimensional model to evaluate the software projects performance, called QEST (**Q**uality factor + **E**conomic, **S**ocial and **T**echnical dimensions). In this study they use geometric concepts to define how to evaluate the performance, where the dimensions are represented by the factors **E**, **S** and **T** and adjusted by a factor **Q** based on qualitative perceptions.

3.5 Concluding Remarks

In fact, there is no direct and safe response for the question: **What is the best software productivity metric?** The simplest and most commonly used metric is the SLOC and its derivations, but these are admittedly problematic for this purpose. Moreover, value-based metrics, which involve assessment of productivity in many dimensions, show a good alignment with the idea of producing value and are gaining popularity today, but are more complex, less common and are still maturing.

According to Boehm [49, 50] quantifying the outcome of which is produced in a software project should not be reduced to number and complexity of requirements or code, but to the product value for the stakeholders. In this thought, the productivity metrics based on value, which are based on multiple dimensions, and which correspond to what the stakeholders consider as the value being produced, is the most complete and reliable to the organizational core business perspective.

Morasca and Russo [43], for example, concluded that using both Function Point and SLOC as a measure of output, which are both interesting depending on the perspective of analysis used. According to them, different stakeholders perceive the result of the work differently and evaluate the *outcomes* of a project differently. This point reinforces the idea that to make use of a good metric of productivity is essential (1) understanding the different *outcomes*; (2) understanding the importance of each of these to the stakeholders; (3) knowning the perspective which the productivity is being evaluated. The item (3), in particular, is extremely important because the same project can be extremely productive from a technical viewpoint (e.g., producing many of FP or SLOC per hour), but a disaster in the viewpoint of the business (if it cannot have a good profit).

Many divergences exist over what is produced during the project and what should be measured. Even some intangibles aspects such as quality and value of what is being produced as a result of a project, become it difficult to define exactly "What is productivity in software projects". In this context, some argue that to measure, in some way, is better than simply not measure at all [14, 51]. However, Austin [52], alerts about the risks of measuring dimensions that are not strongly linked to the phenomenon being observed. He says there is a strong possibility that by doing this, the performance of the process or phenomenon being measured will worsen, despite the contrary opinion of the outcome metrics. This phenomenon was widely observed by the author in different types of organizations, receiving the name of **Dysfunction**. In particular, productivity measurement initiatives presents a great availability to emergence and deployment of dysfunction. For this reason it is suggested to be very carefully while set up a productivity measurement model in software organizations.

The realization of the dimensions of value in software development is not a easy task, several of them are difficult and costly to monitor or are simply intangible. In addition, each organization has a singular work way and different viewpoint about the value of *outputs* produced as a result of a software project. Even this viewpoint varies within the organization, depending on the level of those involved in the evaluation of productivity measures. Therefore, there is no way to define a universal model for measuring productivity to any type of software organization and projects. Each type of organization must define its own model for productivity measurement, which can even vary depending on the types of projects that runs it.

4 Related Works

Maxwell et al. [53] developed a research of analyze the European Space Agency software development projects database with the purpose of identify the main factors influencing the productivity as well as determine the best metric to measure productivity. In this article, the authors present a review of previous research on productivity metrics too, but because the main focus was not this, they mention only 14 publications. To determine the best metric they make a comparison between SLOC and Process Productivity [54] and conclude that SLOC is better.

Scacchi [4] also examines the state of the art in productivity measurement and indicates challenges involved in this practice. As Maxwell et al., he shows the factors which influence productivity. Finally, he suggested the construction of a simulation and modeling system for software development productivity based on knowledge.

Both works did a general review on productivity, including metrics and factors that influence the productivity. Our work is different because it operates with greater depth on the specific problem of measuring productivity. In this work we make a survey of historical quotes on the subject, analyzing the most common uses for each period and tried to identify future trends. Furthermore, we defined a categorization of productivity metrics in order to organize the diversity in this area and facilitate the exploration of the theme.

5 Future Directions

More effort needs to be done towards the definition and use of multidimensional models for measuring productivity. There are several evident problems arising from the use of traditional models that considers only a single metric, or more specifically to assess a single dimension of the production of software [52], however many reports and conclusions are made based on the use of these models. To change the state of the art in software productivity, the first step is to develop specifically the expertise in measuring productivity. For this, the use and development of more modern approaches, which seek to achieve the real dimensions of value produced in software projects are needed.

In special, a well defined process, which takes into account the difficulties of measuring the productivity and reflects the best practices in measurement programs, can help the organization to maximize the chances of success in defining and adopting a model of effective productivity measurement. Based on the approach of Hubbard [51], which argues that anything can be measured, on the recommendations and care cited by Austin [52] and on productivity measurement literature review, we proposes the development of a process to support the definition of productivity measures in software projects. The result of applying this process produces a model of productivity measurement that takes into account the most relevant dimensions of value to the organization, based on the vision of interested in monitoring and evaluating the productivity of projects.

Finally, a research area with good potential to produce good results on the issue of productivity measurement is the research, refinement and adaptation of methods and concepts already mature in other areas such as Administration, Production Engineering and Economics, to Software Engineering. Specifically, in the area of economics concepts and models related to the productivity measurement are more mature and well documented. For example, there are clear definitions of types of different products with different objectives, such as *Capital Productivity* and *Labour Productivity* can be better interpreted in the software development environments.

References

1. DeMarco, T.: Controlling software projects : management, measurement and estimation. Yourdon Press (1982)
2. Jones, C.: Measuring programming quality and productivity. IBM System Journal **17**(1) (1978) 115–125
3. Boehm, B.: Improving software productivity. Computer **20**(9) (September 1987) 43–57
4. Scacchi, W.: Understanding software productivity: a comparative empirical review. System Sciences, 1989. Vol.II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on **2** (Jan 1989) 969–977 vol.2
5. Naur, P., Randell, B., eds.: Software Engineering: Report of a conference sponsored by the NATO Science Committee. Scientific Affairs Division, NATO, Garmisch, Germany (1969)
6. Farr, L., Zagorski, H.J.: Factors that affect the cost of computer programming: A quantitative analysis. Technical Report TM-1447/001/00, System Development Corporation, Santa Monica, California (August 1964)
7. Farr, L., Nanus, B.: Factors that affect the cost of computer programming. Technical Report TM-1447/000/02, System Development Corporation, Santa Monica, California (July 1964)
8. Farr, L., Zagorski, H.J.: A summary of an analysis of computer programming cost factors. Technical Report TM-1447/002/00, System Development Corporation, Santa Monica, California (January 1965)
9. Nelson, E.A.: Management handbook for the estimation of computer programming costs. Technical Report AD-A648750, System Development Corporation (Octuber 1966)
10. Fleishman, T.: Current results from the analysis of cost data for computer programming. Technical Report TM-3026/000/01, System Development Corporation, Santa Monica, California (July 1966)
11. LaBolle, V.: Statistical analysis of computer programming costs. In: SIGCPR '66: Proceedings of the fourth SIGCPR conference on Computer personnel research, New York, NY, USA, ACM (1966) 29–38
12. Pietrasanta, A.M.: Current methodological research. In: Proceedings of the 1968 23rd ACM national conference, New York, NY, USA, ACM (1968) 341–346
13. Aron, J.D.: Estimating resources for large systems. In: In NATO Conference Report on Software Engineering Techniques, Rome, Italy, Brussels: NATO Science Committee (October 1969) 68–79
14. Gilb, T.: Software Metrics. Winthrop Publishers, Cambridge, Mass (1977)
15. Farr, L., LaBolle, V., Willmorth, N.E.: Planning guide for computer program development. Technical Report TM-2314/000/00, System Development Corporation, Santa Monica, California (May 1965)
16. Walston, C.E., Felix, C.P.: A method of programming measurement and estimation. IBM System Journal **16**(1) (1977) 54–65

17. Lawrence, M.J.: Programming methodology, organizational environment, and programming productivity. Journal of Systems and Software **2**(3) (September 1981) 257–269
18. Bailey, J.W., Basili, V.R.: A meta-model for software development resource expenditures. In: ICSE '81: Proceedings of the 5th international conference on Software engineering, Piscataway, NJ, USA, IEEE Press (1981) 107–116
19. Boehm, B.: Software Engineering Economics. Prentice Hall PTR, Upper Saddle River, NJ, USA (1981)
20. Boehm, B., Elwell, J.F., Pyster, A.B., Stuckle, E.D., Williams, R.D.: The TRW software productivity system. In: ICSE '82: Proceedings of the 6th international conference on Software engineering. (1982) 148–156
21. Vosburgh, J., Curtis, B., Wolverton, R., Albert, B., Malec, H., Hoben, S., Liu, Y.: Productivity factors and programming environments. In: ICSE '84: Proceedings of the 7th international conference on Software engineering, Piscataway, NJ, USA, IEEE Press (1984) 143–152
22. Grady, R.B., Caswell, D.L.: Software Metrics: Establishing a Company-Wide Program. Prentice Hall PTR (1987)
23. Card, D.N., McGarry, F.E., Page, G.T.: Evaluating software engineering technologies. IEEE Trans. Softw. Eng. **13**(7) (1987) 845–851
24. MacCormack, A., Kemerer, C.F., Cusumano, M., Crandall, B.: Trade-offs between productivity and quality in selecting software development practices. IEEE Softw. **20**(5) (2003) 78–85
25. Reifer, D.J.: Let the numbers do the talking. CrossTalk The Journal of Defense Engineering (March 2002) 1–8
26. Moses, J., Farrow, M., Parrington, N., Smith, P.: A productivity benchmarking case study using bayesian credible intervals. Software Quality Control **14**(1) (2006) 37–52
27. Albrecht, A.: Measuring application development productivity. In Press, I.B.M., ed.: IBM Application Development Symp. (October 1979) 83–92
28. Behrens, C.A.: Measuring the productivity of computer systems development activities with function points. IEEE Trans. Softw. Eng. **9**(6) (1983) 648–652
29. Banker, R.D., Datar, S.M., Kemerer, C.F.: A model to evaluate variables impacting the productivity of software maintenance projects. Manage. Sci. **37**(1) (1991) 1–18
30. Parkan, C., Lam, K., Hang, G.: Operational competitiveness analysis on software development. Journal of the Operational Research Society **48**(9) (September 1997) 892–905
31. Jones, C.: Applied software measurement: assuring productivity and quality. McGraw-Hill, Inc., New York, NY, USA (1991)
32. Symons, C.R.: Software sizing and estimating: Mk II FPA (Function Point Analysis). John Wiley & Sons, Inc., New York, NY, USA (1991)
33. Maxwell, K.D., Forselius, P.: Benchmarking software-development productivity. IEEE Softw. **17**(1) (2000) 80–88
34. Arnold, M., Pedross, P.: Software size measurement and productivity rating in a large-scale software development department. Software Engineering, 1998. Proceedings of the 1998 International Conference on (Apr 1998) 490–493
35. Anda, B.C.D., Dreiem, H., Sjøberg, D.I.K., Jørgensen, M.: Estimating software development effort based on use cases-experiences from industry. In: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, London, UK, Springer-Verlag (2001) 487–502
36. Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S., Maegawa, Y.: Estimating effort by use case points: method, tool and case study. In: Software Metrics, 2004. Proceedings. 10th International Symposium on. (Sept. 2004) 292–299

37. Antoniol, G., Fiutem, R., Lokan, C.: Object-oriented function points: An empirical validation. Empirical Softw. Engg. **8**(3) (2003) 225–254
38. Cantone, G., Pace, D., Calavaro, G.: Applying function point to unified modeling language: conversion model and pilot study. In: Software Metrics, 2004. Proceedings. 10th International Symposium on. (Sept. 2004) 280–291
39. Symons, C.R.: Function point analysis: difficulties and improvements. Software Engineering, IEEE Transactions on **14**(1) (January 1988) 2–11
40. Ejiogu, L.O.: Software engineering with formal metrics. QED Information Sciences, Inc., Wellesley, MA, USA (1991)
41. III, V.C.: Change-points: a proposal for software productivity measurement. J. Syst. Softw. **31**(1) (1995) 71–91
42. Moser, S., Nierstrasz, O.: The effect of object-oriented frameworks on developer productivity. Computer **29**(9) (1996) 45–51
43. Morasca, S., Russo, G.: An empirical study of software productivity. In: COMPSAC '01: Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development, Washington, DC, USA, IEEE Computer Society (2001) 317–322
44. Duncan, A.S.: Software development productivity tools and metrics. In: ICSE '88: Proceedings of the 10th international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1988) 41–48
45. Anselmo, D., Ledgard, H.: Measuring productivity in the software industry. Commun. ACM **46**(11) (2003) 121–125
46. Hastings, T.E., Sajeev, A.S.M.: A vector-based approach to software size measurement and effort estimation. IEEE Trans. Softw. Eng. **27**(4) (2001) 337–350
47. Stensrud, E., Myrtveit, I.: Identifying high performance erp projects. IEEE Trans. Softw. Eng. **29**(5) (2003) 398–416
48. L.Buglione, A.Abran: Multidimensional software performance measurement models: A tetrahedron-based design. In R.Dumke, A.Abran, eds.: Software Measurement: Current Trends in Research and Practice, Deutscher Universitats Verlag GmbH (1999) 93–107
49. Boehm, B.: Value-based software engineering: reinventing. SIGSOFT Softw. Eng. Notes **28**(2) (2003) 3
50. Boehm, B., Huang, L.G.: Value-based software engineering: A case study. Computer **36**(3) (2003) 33–41
51. Hubbard, D.W.: How to Measure Anything: Finding the Value of "Intangibles" in Business. 1 edition edn. John Wiley & Sons, Inc. (August 2007)
52. Austin, R.: Measuring and Managing Performance in Organizations. Dorset House Publishing Company (1996)
53. Maxwell, K.D., Wassenhove, L.V., Dutta, S.: Software development productivity of european space, military, and industrial applications. IEEE Trans. Softw. Eng. **22**(10) (1996) 706–718
54. Putnam, L.H., Myers, W.: Measures for Excellence: Reliable Software on Time, within Budget. Prentice Hall Professional Technical Reference (1991)