

# Defining and measuring software development output

## A light at the end of the tunnel for an essential problem

Peleg Yiftachel

Department of Computer Science  
University of Haifa, Haifa, Israel  
[peleg@cs.haifa.ac.il](mailto:peleg@cs.haifa.ac.il)

Irit Hadar

Department of Management Information Systems  
University of Haifa, Haifa, Israel  
[hadari@mis.haifa.ac.il](mailto:hadari@mis.haifa.ac.il)

The workshop's website raises the interesting question of whether the original "No Silver Bullet" article [2] was optimistic, pessimistic, or sadly, realistic. This position paper follows the direction presented in David Harel's paper [3]: although we *agree with many of its individual points, the paper present a far gloomier assessment of the situation than seems appropriate*. In other words, we believe Brooks' paper, although raising important and valuable points, to be too pessimistic in its conclusions.

Bearing in mind that 20 years after Brooks's paper, none of the four essential problems are solved, we believe that there is a light at the end of the tunnel for successfully coping with at least some of their implications. This paper presents one of them, creating a new viewpoint for one aspect of the management problem caused by the essential complexity problem. We argue that this aspect is solvable.

One of the critical problems in software engineering is the problem of defining and measuring the output of the software development process. We analyze this problem as an implication of the complexity problem. As Brooks described it: "Software entities are more complex for their size than perhaps any other human construct because **no two parts are alike**. [...]" Likewise, a scaling-up of a software entity is not merely a repetition of the same elements in larger sizes, it is necessarily an increase in the number of different elements. In most cases elements interact with each other in some nonlinear fashion and the complexity of the whole increases much more than linearly." Later he adds: "Not only technical problems, but **management problems** as well come from the complexity."

Like Brooks, we believe that management problems come from complexity, among others. Specifically, we think that one of the sources of the management problems caused by complexity is the inability to have a consistent and measurable definition for the output of a software development process. Finding a way to measure **how much** software we produce will enable analyzing and comparing the productivity of different manufacturers, projects, development strategies, etc. Moreover, it will help cope with the painful problem of cost estimation. An appropriate definition of output is essential for advancing the discipline of software development management, based on scientific, economic foundations. For example, it would enable formulating the demand, production, and cost functions.

One root of the difficulty in measuring software output is that SE deals with *development* of a single prototype

rather than mass *production* of identical goods. Thus, it is meaningless to count the number of finished goods in order to examine output and productivity. A required direction for coping with this problem is measuring the atoms that software is composed of. In order to do so we need to decide what these atoms are. A basic and common approach is the notorious LOC (lines of code) measure. A more advanced approach is the function points (FP) measure [1], giving meaning to the volume of the software functionality, which is indeed a significant factor of the software development output. The problem with these approaches is that they do not take into account the quality of the product, as noted by Sommerville [5, chapter 26]. For example, if two projects produced 100 FPs in a certain period, the FP method will consider them to have the same productivity even if the quality of one is significantly higher than that of the other.

It seems that the reason that such an essential problem is not yet resolved lies in the fact that quality is defined specifically for each product. Although the general quality factors and sub-factors are defined by standards, e.g. of ISO and IEEE, there is an inherent difficulty to compare, for example, the quality of a computer game to the quality of satellite navigation system. Knowing how to represent a product's quality within the measure of a development process's output would enable us to cope better with this challenge.

In past years, much effort has been invested in the issue of software quality. Some effort deals with quality's economic aspects. The COQUALMO model approach [6] recognizes the economic importance of quality and presents relationships between costs, schedule, and quality. We believe this to be a step in the right direction of acknowledging quality as part of software development's output.

In previous research, we demonstrated a practical way to define the output of a software development process, with quality as an inherent part of this definition [7, 8]. We demonstrated how this quality-based definition is beneficial when coping with allocation of resources among software development phases – a critical management problem derived from the complexity problem as presented by Brooks. Our basic idea is that a SW product is defined in a bi-level manner as a collection of features and their qualities, which is the output produced by the manufacturer and consumed by the customer. The first level of this definition, *internal quality*, is associated with the quality of the different phases artifacts, while the second level, *external*

quality, is based on the quality of the SW product from the customer's point of view.

Table 1 shows a matrix representing a SW development's output in the form of the SW product's features and their qualities, as viewed by the customer. This matrix incorporates both the notion of the *amount of functionality* delivered and the *quality* at which various functions are provided. The columns represent different functional features of the SW product. These features can be defined according to the *FP* method, *Object Point* method, or *customer stories*. The rows correspond to external quality characteristics, represented here according to ISO 9126 quality factors: functionality, reliability, usability, efficiency, maintainability, and portability [4].

Quality Factors	Feature 1		Feature 2			Feature 3	Feature 4		...
	Feature 1.1	Feature 1.2	Feature 2.1	Feature 2.2	Feature 2.3	Feature 2.4	Feature 4.1	Feature 4.2	Feature 4.4
Functionality									
Reliability									
Usability			Usability defined for entire feature only			NA			
Portability			Portability is defined for entire SW only						
Efficiency			NA	NA			NA		
Maintainability									

Table 1: The software development output matrix

Our research approach utilizes and expands this definition in order to pose a collection of optimization problems that model the overall software resource allocation problem. Additionally, we characterized the model's components based on empirical studies. We view this and other similar approaches as being meaningful, if yet small, steps in the right direction for better understanding of and improving the management and economic aspects of software development.

The research efforts thus far may appear to rely on the (naïve) assumption that there is a silver bullet for solving the resources allocation problem. However, on the way, we encountered many interesting and beneficial findings (Cf. [7, 8]), which by themselves were worth the trip. Thus, our conclusion is that even if Brooks is right in concluding that there may probably not exist a single, pure silver bullet for solving essential problems of SE, we believe we should all act as if it does, since the process of looking for this silver bullet will at the minimum advance and improve our field.

#### Acknowledgments:

We would like to thank Professor Daniel Berry, Professor Dan Peled, and Mr. Regev Porat for their support and helpful reviews during the creation of this position paper.

#### References:

- [1] Boehm, B., Horowitz, E., Madachy, R. Reifer, D., Clark, B.K., Steece, B., Winsor, A.B, Chulani, S., Chris, A. Resources Software Cost Estimation with COCOMO II. Prentice-Hall, New Jersey, 2000.
- [2] Brooks, F. P., "No Silver Bullet' Refired," in *The Mythical Man-Month*, anniversary edition, Addison Wesley Longman, 1995, pp. 207-226.
- [3] Harel, D. "Biting the Silver Bullet - Toward a Brighter Future for System Development", *IEEE Computer* Vol 25 No 1, January 1992, pp. 8-20
- [4] ISO/IEC TR 9126: Software engineering –Product quality, 19-12-2000.
- [5] Sommerville I., *Software Engineering*, 7th ed., London, Pearson Education Limited, 2005.
- [6] Steece, B., Chulani, S. and Boehm B. "Determining software quality using COQUALMO," in *Case Studies in Reliability and Maintenance*, W. Blischke and D. Murthy, Eds., Sidney, Wiley, 2002.
- [7] Yiftachel, P., *Resource Allocation among Software Development Phases*. MSc. Thesis, University of Haifa, 2006.
- [8] Yiftachel P., Peled D., Hadar I. and Goldwasser D., "Resource Allocation among Development Phases: An Economic Approach," presented at the *Eighth International Workshop on Economics-Driven Software Engineering Research*, Shanghai, 2006.