

WITHDRAWN

Recognized as an
American National Standard (ANSI)

IEEE Std 1045-1992

IEEE Standard for Software Productivity Metrics

Sponsor

**Software Engineering Standards Subcommittee
of the
Technical Committee on Software Engineering
of the
IEEE Computer Society**

Approved September 17, 1992

IEEE Standards Board

Approved March 22, 1993

American National Standards Institute

Abstract: A consistent way to measure the elements that go into computing software productivity is defined. Software productivity metrics terminology are given to ensure an understanding of measurement data for both source code and document production. Although this standard prescribes measurements to characterize the software process, it does not establish software productivity norms, nor does it recommend productivity measurements as a method to evaluate software projects or software developers. This standard does not measure the quality of software. This standard does not claim to improve productivity, only to measure it. The goal of this standard is for a better understanding of the software process, which may lend insight to improving it.

Keywords: attribute, primitive, productivity ratio, source statement, staff-hour

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1993 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1993. Printed in the United States of America

ISBN 1-55937-258-3

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

WITHDRAWN

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

WITHDRAWN

Introduction

(This introduction is not a part of IEEE Std 1045-1992, IEEE Standard for Software Productivity Metrics.)

This introduction is intended to provide the reader with some background into the rationale used to develop the standard. This information is being provided to aid in the understanding and usage of the standard. The introduction is nonbinding.

This standard defines a framework for measuring and reporting software productivity. It focuses on definitions of how to measure software productivity and what to report when giving productivity results. It is meant for those who want to measure the productivity of the software process in order to create code and documentation products.

Past software productivity metrics have not proven as useful as desired to provide insight into the software process. Although there is an accumulation of more than 20 years of data, consistent productivity indicators for software development have not emerged from this information. The problem is not as much the fault of the metrics being used as it is the inaccuracy and incompleteness of the data being collected.

The definition of productivity states that it is the ratio of a unit of output to a unit of input used to produce the output. For this relationship to be useful for software, the data used in it must be accurate and complete. For instance, reported software productivity of 5000 lines of source code per year leaves many questions unanswered, *What is a line of source code? How long, in work hours, was the year? What activities were included? Whose effort was counted?*

Interpreting productivity based on a single number leaves much unknown about the process being measured. Without knowing the scope and characteristics of the process measured, or the precision of the data used in the calculations, the resulting productivity values are inconclusive.

The goal of this standard is to build a foundation to accurately measure software productivity. This is done through a set of precisely defined units of measure. However, not all software processes lend themselves to precise measurement. Software development is a new and rapidly evolving field, and it is strongly influenced by the variability of the people who build the software. In those situations where precise measurement definitions are not possible, this standard requests that descriptions of the processes used and the measurements taken be done in a specified format.

The intention of the standard is to formalize the presentation of productivity data so that it is useful to anyone wishing to improve the software process. This standard is not an end in itself. Instead, it is the beginning of increased precision in collecting and reporting software productivity data. The hope is that this will lead to an improved understanding of the software development process and to improved productivity metrics.

WITHDRAWN

Participants

At the time this standard was completed, the Software Productivity Metrics Working Group had the following membership:

Robert N. Sulgrove, *Chair*
Eleanor Antreassian, *Past Chair*

Christine H. Smith, *Co-chair*
Nicholas L. Marselos, *Editor*

Bakul Banerjee
Stephen E. Blake
Thomas P. Bowen
David N. Card
Thomas J. Carlton
Deborah Caswell
Mike Demshki
Sherman Eagles
Ruth S. Euler
Michael Evangelist
Al Freund
James T. Fritsch

John E. Gaffney, Jr.
Stuart Glickman
Lennor Gresham
Stuart Jeans
Robert W. Judge
Lawrence King
Thomas M. Kurihara
Arnold W. Kwong
Fred Lau
Chi Yun Lin
Denis C. Meredith
Lois J. Morton
Andrew Najberg

Jainendra K. Navlakha
Dennis E. Nickle
Richard Reese
Julian Roberts
Brian Sakai
Sylvia Shiroyama
Paul Stevens
Wolfgang B. Strigel
Leonard L. Tripp
Scott A. Whitmire
Ron Willis
Paul Wolfgang

Contributors

The following individuals also contributed to the development of the standard:

William W. Agresti
Lowell Jay Arthur
Jeff A. Aune
Victor R. Basili
Mordechai Ben-Menachem
Victor G. Berez
Robert C. Birss
Bob Bisschoff
Barry Boehm
George Bozoki
Fred Burke
Neva Carlson
Sally Cheung
Rutherford Cooke
Charles D'Argenio
James B. Dolkas
Carl Einar Dragstedt
Christof Ebert
Violet Foldes
Andrew S. Fortunak
Robert Fraley
Jack Fried
Jean A. Gilmore
Clell Gladson
J. G. Glynn
Robert Grady
Dan Grigore
Amal Gupta
Nash Hair
H. D. Hall

Jack Harrington
Warren Harrison
Bruce Heaton
Francis B. Herr
Herman Hess
Geoffrey W. Higgin
John W. Horch
David Hurst
Randall W. Jensen
Bud Jones
T. Capers Jones
Bill Junk
Motti Y. Klein
Phil Kolton
Steven E. Kreutzer
Walt Kutz
Robert L. Lanphar, Jr.
F. C. Lim
Michael Lyu
Andy Mahindru
Jukka Marijarvi
Phillip C. Marriott
Roger J. Martin
Robert F. Martini
Joseph F. Mathews
Bruce Millar
James Miller
Robert C. Natale
Stephen R. Neuendorf
Ken O'Brien

Randy Paddock
Bruce Parker
Bud W. Pezet
Wes Philp
Robert M. Poston
Lawrence H. Putnam
Donald J. Reifer
Niall Ross
Vince Rupolo
Norman F. Schneidewind
Roger Scholten
David J. Schultz
Suzanne E. Schwab
Carl Seddio
Al Serna
Jean Shaffer
Vincent Y. Shen
Josef Sherif
David M. Siefert
Vijaya K. Srivastava
Edwin J. Summers
David Swinney
Robert C. Tausworthe
C. L. Troyanowski
Dolores Wallace
Richard Werling
John Westergaard
Clyde E. Willis
William Wong
Weider Yu

WITHDRAWN

Sponsoring Organizations

The following persons were on the balloting committee that approved this standard for submission to the IEEE Standards Board:

M. Amaya	C. Kemerer	H. Schaefer
B. Banerjee	R. Kessler	N. Schneidewind
L. Beltracchi	L. King	G. Schumacher
M. Ben-Menache	T. Kurihara	C. Seddio
R. Birss	L. Lam	R. Shillato
S. Blake	R. Lamb	S. Shiroyama
W. Boll	J. Lane	D. Siefert
R. Both	J. Lawrence	C. Smith
F. Buckley	F.C. Lim	V. Srivastava
D. Card	B. Livson	W. Strigel
N. Carlson	D. Look	R. Sulgrove
W. Chung	M. Lyu	W. Thetford
F. Coallier	J. Maayan	G. Trebble
P. Daggett	H. Mains	L. Tripp
B. Derganc	N. Marselos	M. Updike
C. Ebert	R. Martin	R. Van Scoy
R. Euler	T. Matsubara	D. Wallace
W. Eventoff	I. Mazza	J. Walz
F. Frati	L. Miller	S. Whitmire
R. Fries	A. Najberg	P. Work
J. Gaffney, Jr.	J. Navlakha	A. Yonda
Y. Gershkovich	D. Nickle	W. Yu
A. Godin	P. Petersen	L. Heselton
D. Gustafson	S. Redwine	C. Hu
W. Harrison	R. Reese	W. Perry
W. Hefley	D. Reifer	I. Trandafir
P. Hinds	B. Sakai	A. Wainberg
J. Horch	R. San Roman	P. Zoll
	J. Sanz	

When the IEEE Standards Board approved this standard on Sept. 17, 1992, it had the following membership:

Marco W. Migliaro, Chair **Donald C. Loughry, Vice Chair**
Andrew G. Salem, Secretary

Dennis Bodson	Donald N. Heirman	T. Don Michael*
Paul L. Borrill	Ben C. Johnson	John L. Rankine
Clyde Camp	Walter J. Karplus	Wallace S. Read
Donald C. Fleckenstein	Ivor N. Knight	Ronald H. Reimer
Jay Forster*	Joseph Koepfinger*	Gary S. Robinson
David F. Franklin	Irving Kolodny	Martin V. Schneider
Ramiro Garcia	D. N. "Jim" Logothetis	Terrance R. Whitemore
Thomas L. Hannan	Lawrence V. McCall	Donald W. Zipse

*Member Emeritus

Also included are the following nonvoting IEEE Standards Board liaisons:

Satish K. Aggarwal
James Beall
Richard B. Engelman
David E. Soffrin
Stanley Warshaw

Rachel Auslander
IEEE Standards Project Editor

WITHDRAWN

Contents

CLAUSE	PAGE
1. Overview.....	1
1.1 Scope.....	1
1.2 Terminology.....	2
1.3 Audience	2
2. References.....	2
3. Definitions.....	3
4. Software productivity metrics.....	4
5. Output primitives	5
5.1 Source statement output primitives.....	5
5.2 Function point output primitive	9
5.3 Document output primitives	9
6. Input primitive	11
6.1 Staff-hour input primitive	11
6.2 Staff-hour attribute.....	12
6.3 Activities	12
7. Relationships.....	12
7.1 Productivity ratios.....	12
7.2 Output-to-output ratios.....	16
7.3 Input-to-input ratios	19
8. Characteristics.....	19
8.1 Project characteristics	20
8.2 Management characteristics.....	22
8.3 Product characteristics	22
ANNEXES	
Annex A Sample metrics data collection summary list.....	25
Annex B Characteristics data collection form.....	27
Annex C Bibliography	28
Annex D Software counting relationships	29

IEEE Standard for Software Productivity Metrics

1. Overview

This standard describes the data collection process and calculations for measuring software productivity. This standard is divided into eight clauses. Clause 1 provides the scope of this standard. Clause 2 lists references to other standards that are useful in applying this standard. Clause 3 provides an abbreviated set of definitions and acronyms defined more fully in the standard, but provided here as a quick reference. Clause 4 provides an introduction to the standard's measurement approach. Clause 5 describes data collection for measuring output. Clause 6 describes data collection for measuring input. Clause 7 describes productivity results represented by ratios of outputs to inputs. Clause 8 provides a method to capture the characteristics of the software process to better understand their effects on productivity results.

This standard also contains four annexes. These provide additional information for understanding and using this standard, but they are not part of the standard. Annex A is a data collection form for recording quantitative metrics data defined in this standard, and Annex B is a data collection form for recording the characteristics data defined in this standard. Annex C contains a bibliography of some of the many references used in developing this standard. Annex D describes a counting method for the relationships defined in the standard.

1.1 Scope

Measuring software productivity is similar to measuring other forms of productivity; that is, it is measured as the ratio of units of output divided by units of input. The inputs consist of the effort expended to produce a given output. For software, the tangible outputs are source code and documentation. Of course the product of software is a process, an algorithm that drives a computer to do work. Unfortunately, understanding of the functionality incorporated into a software product is still rudimentary. Only by understanding what can be measured will it be possible to understand the essence of the software process.

Toward this end, this document standardizes software productivity metrics terminology to ensure an understanding of measurement data for both code and documentation production. It defines a set of units to measure the output products and input effort.

The lowest level of measurement defined in this standard is called a *primitive*. The output primitives measured are software source statements, documentation pages, and, optionally, function points. The input primitives measure the efforts of those developing the software products. The capacity and capability of automated support tools are not directly measured by this standard, but are indirectly measured by the improvements in the productivity of the people who use them.

This standard prescribes measurements to characterize the software process, and in doing so gives insight for improving it. This standard does not establish software productivity norms, nor does it recommend productivity measurements as a method to evaluate software projects or software developers.

Although the overall value of a product cannot be separated from the quality of what is produced, this standard does not measure the quality of software. The issue of measuring quality is beyond the scope of this standard, because it covers a different aspect of the software process. Nevertheless, productivity metrics should be interpreted in the context of the overall quality of the product. It is left to the user of this standard to look to other standards covering quality metrics for that information.

The definition of productivity as the ratio of product to effort was selected in this standard because it is more stable than value relationships like product to monetary units or production to sales. The effects of inflation on the value of monetary units, or the caprice of the marketplace on sales, cause these measures to vary unpredictably. As a result, they do not offer a stable measure over time. However, the quantity of the product and the effort in hours expended to produce it are more consistent measures, and over time these measures will provide a better gauge of software productivity. Users may wish to translate productivity into monetary equivalents, but results shall be reported in the units specified in this standard.

The metrics in this standard apply equally well to new development and to the enhancement or maintenance of an existing software product. Subsequent releases or changes to a released or delivered software product should be viewed as a new product for the purpose of applying these metrics.

This standard defines a consistent way to measure the elements that go into computing software productivity. A consistent measurement process will lead to a better understanding of the software development process, and a better understanding will lead to improvement. This standard does not claim to improve productivity, only to measure it.

1.2 Terminology

The words *shall*, *must*, and words in the imperative form identify the mandatory (essential) material within this standard. The words *should* and *may* identify optional (conditional) material. As with other IEEE Software Engineering Standards, the terminology in this document is based on the IEEE Std 610.12-1990.¹ To avoid inconsistency when the glossary is revised, the definitions are not repeated in this document. New terms and modified definitions as applied in this standard are included in clause 3.

1.3 Audience

This standard should be of interest to software development managers, software productivity and process assessment personnel, quality assurance personnel, software engineering improvement personnel, and researchers.

2. References

This standard shall be used in conjunction with the following publications:

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology (ANSI).

IEEE Std 1074-1991, IEEE Standard for Developing Software Life Cycle Processes (ANSI).

¹Information on references can be found in clause 2.

3. Definitions

This clause contains key terms as they are used in this standard.

3.1 activities: Events in the software life cycle for which effort data is collected and reported.

3.2 added source statements: The count of source statements that were created specifically for the software product.

3.3 attributes: Measurable characteristics of a primitive.

3.4 characteristics: Those inherent factors of software development that may have a significant impact on productivity.

3.5 comment source statements: Source statements that provide information to people reading the software source code and are ignored by the compiler.

3.6 compiler directive source statements: Source statements that define macros, or labels, or direct the compiler to insert external source statements (for example, an *include* statement), or direct conditional compilation, or are not described by one of the other type attributes.

3.7 data declaration source statements: Source statements that reserve or initialize memory at compilation time.

3.8 deleted source statements: Source statements that are removed or modified from an existing software product as a new product is constructed.

3.9 delivered source statements: Source statements that are incorporated into the product delivered to the customer.

3.10 developed source statements: Source statements that are newly created for, added to, or modified for a software product.

3.11 direct staff-hour: The amount of effort directly expended in creating a specific output product.

3.12 document page count: The total number of nonblank pages contained in a hard-copy document.

3.13 document screen count: The total number of page images for electronically displayed documents.

3.14 documents: Hard copy, screen images, text, and graphics used to convey information to people.

3.15 executable source statements: Source statements that direct the actions of the computer at run time.

3.16 function point: A measure of the delivered software functionality.

3.17 granularity: The depth or level of detail at which data is collected.

3.18 incremental productivity: The productivity computed periodically during development.

3.19 input primitive: The effort to develop software products, expressed in units of staff-hours.

3.20 logical source statements (LSS): Source statements that measure software instructions independently of the physical format in which they appear.

- 3.21 modified source statements:** Original source statements that have been changed.
- 3.22 new source statements:** The sum of the added and modified source statements.
- 3.23 nondelivered source statements:** Source statements that are developed in support of the final product, but not delivered to the customer.
- 3.24 nondeveloped source statements:** Existing source statements that are reused or deleted.
- 3.25 origin attribute:** The classification of software as either developed or nondeveloped.
- 3.26 original source statements:** Source statements that are obtained from an external product.
- 3.27 output primitives:** Primitives that include source statements, function points, and documents.
- 3.28 page size:** The edge-to-edge dimensions of hard-copy documents, or the average characters per line and the number of lines per screen for electronically displayed documents.
- 3.29 physical source statements (PSS):** Source statements that measure the quantity of software in lines of code.
- 3.30 primitive:** The lowest level for which data is collected.
- 3.31 productivity ratio:** The relationship of an output primitive to its corresponding input primitive.
- 3.32 reused source statements:** Unmodified source statements obtained for the product from an external source.
- 3.33 source statements (SS):** The encoded logic of the software product.
- 3.34 staff-hour:** An hour of effort expended by a member of the staff.
- 3.35 support staff-hour:** An hour of effort expended by a member of the staff who does not directly define or create the software product, but acts to assist those who do.
- 3.36 tokens:** The content of a document as characterized by words, ideograms, and graphics.
- 3.37 type attributes:** The classification of each source statement as either executable, data declaration, compiler directive, or comment.
- 3.38 usage attributes:** The classification of software as delivered to a user of the final product, or as nondelivered when created only to support the development process.

4. Software productivity metrics

This standard defines measurement primitives for computing software productivity. A primitive is the lowest level for which data is collected. The output primitives for software are source statements and, optionally, function points. For documentation, the output primitive is the page. The input primitive is the staff-hour.

Primitives are categorized by attributes. An attribute is a measurable characteristic of a primitive. This standard requires that all attributes be measured for each primitive. For example, the source statement primitive may be either new or reused, and it may also be either delivered or nondelivered. Each of these attributes is a different variation of the primitive. Thus, a new delivered source statement differs from a new nondeliv-

ered source statement. By using this scheme of primitives and attributes, the elements of productivity are consistently categorized.

Because productivity measurements are only as valid as the primitive data used to calculate them, accuracy and consistency of the data used are essential for the results to be meaningful. The degree of granularity, which is the depth or level of detail at which data is collected, is important in determining the precision and consistency of the data.

Data of fine granularity is obtained when collecting data by staff-hour for each hour worked. Tracking the effort expended at this level reduces the potential for large errors. Estimates, such as those made at the end of a project by best guess approximation of effort, are an example of very coarse granularity. In this situation, large errors may be introduced into the productivity results.

This standard requires the use of the finest degree of granularity obtainable in collecting software productivity data. For the results of this standard to be most useful, the measuring process used shall be both accurate and detailed. Therefore, recording input effort as it is expended is the only acceptable method for this standard.

5. Output primitives

Three categories of output primitives are defined in this standard: source statements, function points, and documents. Source statements are a fundamental part of software, representing the encoded logic of the software product. Function points are a measure of the functional content of software. Production of documents is a significant component of the effort expended in software development, and measuring productivity for it is important. This standard requires reporting productivity results for source statements and documents. The use of function points is optional but recommended.

5.1 Source statement output primitives

This clause defines methods for counting software statements, and methods for categorizing the nature, origin, and disposition of the software.

Two methods are used to count software source statements: the logical source statement (LSS) method, which counts instructions, and the physical source statement (PSS) method, which counts lines of code. Logical source statement counting is the preferred method for this standard. Physical source statement counting is included in this standard because it is commonly found in published data.

Software is categorized by a set of attributes that describe it. Each source statement primitive is defined by three attribute qualifiers. The first of these is the type attribute that classifies each source statement as either executable, data declaration, compiler directive, or comment. The second is the origin attribute that classifies software as either developed or nondeveloped. Developed software is created new, or modified from existing software, for this product. Nondeveloped software is reused or deleted from the existing software. The last attribute is the usage attribute that identifies whether the software was delivered to a user of the final product, or produced only to support the development process and not delivered to the user.

5.1.1 Source statement counting

Source statements in different programming languages shall be counted separately. The programming language used shall be identified for each count along with its software module name. If two languages are used in the development, for example Pascal and C, the quantity of source statements of each language shall be counted and reported separately. Combining counts of different languages into a single count shall not be permitted.

This standard requires that only those source statements written by people be counted. Other counting practices, such as counting compiler derived instructions, program generator output, reuse library output, or machine instructions may be counted in addition to the people-created source statements provided these counts are kept separate and clearly identified. Counts of object code, executable code, and code that is mechanically generated from source code are not required by this standard. It may be most useful to count source statements at the level that the source is maintained. As a minimum, counting shall be done at product completion, when the product is ready for delivery. In all cases, what is counted shall be specifically stated along with the quantity of software measured.

Source statements that are expanded within a software module, for example, macro expansions, or supplied to the program, for example, by an *include* statement, shall be counted only once for all modules being measured. Source statements that invoke, call, or direct inclusion of other source statements in the module shall be counted each time they are used.

Source statements are counted in one of two ways, as logical source statements (LSS) or as physical source statements (PSS):

- a) *Logical source statements.* The LSS counting method measures the number of software instructions. When a source statement is the count of instructions irrespective of its relationship to lines, the count shall be stated as being in logical source statements.

The LSS method is used to count software statements independently of the physical format in which they appear. For example, if two instructions appear on a single line, the LSS count would be two. If one instruction spans two lines, the LSS count would be one.

An LSS count shall be accompanied by a description of the procedure for counting the source statements. This description shall define the algorithm used to determine the beginning and end of each source statement, and how embedded source statements are counted.

- b) *Physical source statements.* The PSS counting method measures the number of software lines of code. When the end-of-line character or a line of code method is used to count source statements, the count shall be stated as being in physical source statements.

A PSS measurement is line counting where a typical line format is 80 characters. Any deviation from the typical line format shall be noted. A PSS may consist of one LSS, more than one LSS, or part of an LSS.

Blank lines shall be excluded from PSS counts. If desired, blank lines may be counted separately as the number of blank lines.

5.1.2 Source statement attributes

This standard partitions source statements into three attributes: type, origin, and usage. The type attribute consists of executable, data declaration, compiler directive, and comment source statements. The origin attribute consists of developed and nondeveloped source statements. The usage attribute consists of delivered and nondelivered source statements.

Separate counts for each combination of attribute values shall be stated when reporting productivity results. For example, separate counts shall be stated for executable source statements that are developed delivered and for executable source statements that are developed nondelivered.