Measurement Construct - Product Size Growth

Title	Measurement Construct - Product Size Growth
Project/Subject	Professional Development Center
Version	1.0
Status	External Draft 1
Date	5-7-2006
File	Product Size Growth
Company	Info Support b.v.

Einsteinstraat 22[1], 3902 HN Veenendaal, Tel. +31 (0) 318 55 20 20 Fax +31 (0) 318 55 23 55
De Smalle Zijde 39, 3903 LM Veenendaal, Tel. +31 (0) 318 50 11 19 Fax +31 (0) 318 51 83 59
Rabobank Bergh 30.59.52.889, Postbank 47.38.593

History

Version	*Status*	*Date*	*Author*	*Change*
1.0	Draft 1	8-12-2004	Joop Snijder	Creation
1.0	Internal Draft 1	9-2-2004	Joop Snijder	Ready for Review
1.0	External Draft 1	4-3-2004	Joop Snijder	Review Changes (2-3-2005)

Distribution

Version	*Status*	*Date*	*To*
1.0	Internal Draft 1	9-2-2004	Remco Bruns, Dennis Joosten, Marcel de Vries

References

Reference	*Document*
PSM02	John McGarry et al, *"Practical Software Measurement, Objective Information for Decision Makers*", 2002, Addison-Wesley
MMPERF	Robert D. Austin, *"Measuring and Managing Performance in Organizations"*, 1996, Dorset House Publishing Co
ROYCE98	Walker Royce, *"Software Project Management, A Unified Framework"*, 1998, Addison-Wesley

Index

1. Introduction

A measurement construct is a detailed structure that links the base measures to the specified information needs. The measurement construct include specific attributes of software processes and products, such as size, effort, and number of defects. The measurement construct describes how the relevant software attributes are quantified and converted to indicators that provide a basis for decision making.
At each of the three levels of measures – base measures, derived measures and indicators – additional information content is added in the form of rules, models, and decision criteria.

1.1 Definitions

Key terms on software measures and measurement methods have been defined in ISO/IEC 15939 on the basis of the ISO international vocabulary of metrology. The terms used in this document are derived from ISO 15939 and PSM (Practical Software Measurement) [PSM02].

2. Product Size Growth

2.1 Introduction

The most significant way to improve affordability and return on investment (ROI) is usually to produce a product that achieves the design goals with the minimum amount of human-generated source material [ROYCE98].

2.2 Project Application

- Applicable to all sizes and types of projects
- Applicable to all process models
- Measures in this category are usually selected based on domain characteristics

2.3 Universal Function Points

In [ROYCE98] Universal Function Points (UFPs) are defined as useful estimators for language-independent early life-cycle estimates. The basic units of function points are external user inputs, external outputs, internal logical data groups, external data interfaces, and external inquiries. Source lines of code (SLOC) metrics are useful estimators for software after a candidate solution is formulated and an implementation is known. Substantial data have been documented relating SLOC to function points. Some of these results are shown in Table 1. The current language expressiveness of programming languages must be collected.

Language	SLOC/Function point			
	Avg	Median	Low	High
Access	35	38	15	47
ASP	69	62	32	127
C	162	109	33	704
C++	66	53	29	178
Excel	47	46	31	63
HTML	47	53	15	60
Java	62	63	53	77
JavaScript	58	63	42	75
Oracle	30	35	4	217
SQL	40	37	7	110
VBScript	36	34	27	50
VB	47	42	16	158

Table 1: Language expressiveness of some of today's popular[1] languages

2.4 Measurement Information Specification

Information Need Description	
Information Need	Evaluate the size of a software product to appraise the original budget estimate
Information Category	Product Size and Stability

[1] By the time of writing C# was not defined yet.

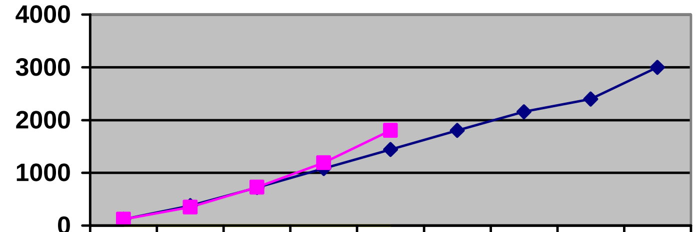
Measurable Concept	
Measurable Concept	Physical size and stability

Entities and Attributes	
Relevant Entities	1. Software development plan or schedule 2. Baselined software source code library
Attributes	1. Function points planned for completion each period 2. Source Lines of Code (SLOC)

Base Measure Specification	
Base Measures	B1. Planned Function Points B2. Actual SLOC
Measurement Methods	B1. Count the cumulative function points that were planned to be completed by the current period. B2. Count the number of lines of code in the currently approved software source code library.
Type of Method	B1. Objective B2. Objective
Scale	B1. Integers from zero to infinity B2. Integers from zero to infinity
Type of Scale	B1. Ratio B2. Ratio
Unit of Measurement	B1. Function Point B2. Source Lines of Code

Derived Measure Specification	
Derived Measure	D1. Universal Function Points (UFPs) D2. Software size growth ratio
Measurement Function	D1. With the use of a language expressiveness table (e.g. 58 SLOC of JavaScript = 1 Universal Function Point). Divide the total number of SLOC with the SLOC per UFP for a specific programming language. D2. Divide the actual UFP by the planned UFP to date

Indicator Specification	
Indicator Description and Sample	1. Trend of software size growth



	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Planned FP	120	380	720	1080	1440	1800	2160	2400	3000
Actual UFP	119	350	725	1190	1800				
Ratio	0,99	0,92	1,01	1,1	1,25				

Analysis Model	Increasing software size growth ratio indicates increasing risk to achieving cost and schedule budgets.
Decision Criteria	Investigate if the software size growth ratio exceeds a variance of 20%.
Indicator Interpretation (sample chart)	The indicator seems to indicate that the project production rate is ahead of schedule. However, after further investigation, it turns out that the actual code count for one configuration item was higher than planned due to missing requirements that were not identified until initial testing. Resource allocations, schedules, budgets, and test schedules and plans are impacted by this unexpected size growth.
Tools Used for Indicator Presentation	Endeavour Project Portal, Microsoft Reporting Services

Indicator Dimension Specification	
Filter	Per Project, per Release, Per CI,
Detail	UFP or SLOCS per Programming Language
Time	Per year, quarter, month, week, day

Data Collection Procedure (for each Base Measure)	
Frequency of Data Collection	B1. From each iteration plan and updated whenever a revision to the plan occurs. B2. Each build
Responsible Individual	Buildmanager
Phase or Activity in which Collected	B1. Elaboration, Construction, Transition B2. Elaboration, Construction, Transition
Verification and Validation	TODO
Tools Used in Data Collection	AutoBuild, Microsoft Team System
Repository for Collected Data	Data Warehouse

Additional Information	
Additional Analysis Guidance	Both code growth and lagging progress in code production are leading indicators of effort and schedule slips, so code growth should be monitored closely throughout development. Analysis of excessive growth may uncover scope changes, unsatisfied technical assumptions regarding reuse or use of commercial products, or underestimation of code counts.
Implemen-tation Considerations	Many projects are working with supplied code libraries, other forms of reusable code, generated code, and modified code as well as new code, these code sources or groupings should be considered as generated code.
Dysfunctional Measurement[2]	• To meet the planned values exactly or closely a project manager may alter the plans afterwards. • To meet the planned values the project may alter the number of source lines of code per UFP. • To meet the planned values, code will be skipped in the Autobuild.

[2] A risk with any metrics activity is dysfunctional measurement, in which participants alter their behavior to optimize something that is being measured, rather than focusing on the real organizational goal.