Tool Description

The *METRICS NEWS* can be ordered directly from the Editorial Office (for address see below).

Editors:

ALAN ABRAN
Professor and Director of the Research Lab. in Software Engineering Management
Quebec-University of Montreal
Departement of Computer Science
C.P. 8888 Succursale Centre-Ville, Montreal, H3C 3P8, Canada
Tel.: *+1-514-987-3000, -89000,* ***Fax:*** *+1-514-987-8477*
Email: *abran.alain@uqam.ca*

MANFRED BUNDSCHUH
Chair of the DASMA
Sander Höhe 5, 51465 Bergisch Gladbach, Germany
Tel.: *+49-2202-35719*
Email: *Bundschuhm@acm.org*
http://www.dasma.de

REINER DUMKE
Professor on Software Engineering
University of Magdeburg, FIN/IVS
Postfach 4120, D-39016 Magdeburg, Germany
Tel.: *+49-391-67-18664,* ***Fax:*** *+49-391-67-12810*
Email: *dumke@ivs.cs.uni-magdeburg.de*

CHRISTOF EBERT
Dr.-Ing. in Computer Science
Alcatel Telecom, Switching Systems Division
Fr. Wellensplein 1, B-2018 Antwerpen, Belgium
Tel.: *+32-3-240-4081,* ***Fax:*** *+32-3-240-9935*
Email: *christof.ebert@alcatel.de*

HORST ZUSE
Dr.-Ing. habil. in Computer Science
Technical University of Berlin, FR 5-3,
Franklinstr. 28/29, D-10587 Berlin, Germany
Tel.: *+49-30-314-73439,* ***Fax:*** *+49-30-314-21103*
Email: *zuse@tubvm.cs.tu-berlin.de*

Editorial Office: Otto-von-Guericke-University of Magdeburg, FIN/IVS, Postfach 4120, 39016 Magdeburg, Germany
Technical Editor: DI Erik Foltin
The journal is published in one volume per year consisting of two numbers.

C A L L F O R P A P E R S

Workshop: Performance Engineering in der Softwareentwicklung
Gastgeber: DeTeCSM GmbH, Benchmarklabor
15. Mai 2000 in Darmstadt

veranstaltet von der:

Otto-von-Guericke Universität Magdeburg
Arbeitsgruppe Softwaretechnik
Arbeitsgruppe Wirtschaftsinformatik

in Zusammenarbeit mit der
- T-Nova Deutsche Telekom Innovationsgesellschaft mbH, Entwicklungszentrum Berlin
- GI-Fachbereich 5: Wirtschaftsinformatik (angefragt)
- GI-Fachgruppe 3.2.1: Messung, Modellierung und Bewertung von Rechensystemen
- GI-Fachgruppe 2.1.10: Software-Messung und Bewertung

THEMENGEBIET PERFORMANCE ENGINEERING:

Einer der kritischsten nicht-funktionalen Qualitätsfaktoren ist die Performance eines Softwaresystems. Performance kann als die Fähigkeit eines Systems verstanden werden, eine gegebene Anzahl von Aufgaben in einer bestimmten Zeitspanne bewältigen zu können. Damit steht die Performance von Softwaresystemen in direkter Beziehung zu der Geschwindigkeit begleitender Geschäftsprozesse. Es obliegt dem Informationsmanagement Kunden-, Zulieferungs- und die eigenen Unternehmensprozesse Performance-gerecht zu integrieren.

Die Kernidee des Performance Engineering besteht vordergründig darin, die Performance eines Informationssystems bereits in den frühen Phasen der Softwareentwicklung zu berücksichtigen. Damit soll die Entwicklung von Softwaresystemen ermöglicht werden, die unter Angabe eines definierten Ressourcenverbrauchs und eines Lastmodells, die von dem späteren Anwender geforderten Leistungsattribute erfüllen können.

In der industriellen Praxis wird jedoch meistens auf pro-aktive Performance-Untersuchungen verzichtet. Häufig wird dieser Qualitätsfaktor erst am Ende der Softwareentwicklung betrachtet. Sollten hier Performance-Probleme mit der gewählten Architektur des IT-Systems festgestellt werden, führt dies zu aufwendigen Tuning-Maßnahmen, der Neubeschaffung leistungsfähigerer Hardware oder zu einem notwendigen Re-Design der Softwareanwendung. Mit derartigen Nachbesserungen sind in jedem Fall Kosten verbunden, die nicht nur die Arbeiten für die Performance-Verbesserungen betreffen, sondern auch Kosten, die beispielsweise durch das nicht zeitgerecht verfügbare System entstehen. Wenngleich ein ständiges Wachstum der Performance neuer Hardwaresysteme zu verzeichnen ist, sind es insbesondere komplexere Anwendungssysteme auf der Basis neuer Technologien, die einer expliziten Beachtung des Performance-Verhaltens während der Entwicklung bedürfen.

EINREICHUNG VON BEITRÄGEN:

Praktiker und Wissenschaftler, die auf dem Gebiet des Performance Engineering und artverwandter Aufgabenbereiche aktiv sind, werden gebeten, entsprechende Beiträge zu folgenden Themenschwerpunkten einzureichen:

Praxisberichte:
- Praxis-/Erfahrungsberichte aus laufenden/abgeschlossenen Projekten
- Aufwandsuntersuchungen für Performance Engineering

Prozesse des Performance-Engineering:
- Risiken nicht performanter Softwaresysteme
- Performance-orientierte Design- und Implementierungstechniken
- Performance-Betrachtungen innerhalb des Entwicklungsprojektes

Modellierung des Performance-Verhaltens innerhalb der Softwareentwicklung:
- Berücksichtigung von "Quality of Service" Eigenschaften
- Gewinnung von Modellvariablen
- Erfahrungen im Umgang mit entsprechenden Tools

Der Umfang der Beiträge sollte 3000 Wörter nicht übersteigen.
Die Formatierungsrichtlinien sind der angegebenen Webseite zu entnehmen.

Die Beiträge sollen in Kurzvorträgen (Vortragslänge ca. 20 Minuten zzgl. 10 Minuten Diskussion) präsentiert werden. Alle angenommenen Beiträge werden in einem Workshopband (Preprint-Reihe der Otto-von-Guericke Universität Magdeburg) veröffentlicht.

Bitte senden Sie ihre Beiträge in einem der Formate doc, rtf, pdf, ps per Email an:

Andre Scholz
Email: ascholz@iti.cs.uni-magdeburg.de

In Ausnahmefällen auch auf dem Postweg an:

Andre Scholz
Otto-von-Guericke-Universität Magdeburg, FIN-ITI
Universitätsplatz 2
39106 Magdeburg

TEILNAHME/ANMELDUNG:

Interessenten werden gebeten, sich unter der folgenden URL-Adresse elektronisch anzumelden:

http://www-wi.cs.uni-magdeburg.de/~ascholz/pe2000

In Ausnahmefällen ist auch eine Anmeldung unter Angabe von Name, Vorname, Firma/ Institution, Anschrift, Tel., Email per Brief / Tel. / Fax über das Kontaktbüro möglich:

Frau Kerstin Lange
Otto-von-Guericke-Universität Magdeburg, FIN-ITI

Universitätsplatz 2
D-39016 Magdeburg
Tel: 0391-6718386 / Fax: 0391-6711216
Email: klange@iti.cs.uni-magdeburg.de

Es wird eine Teilnahmegebühr von 100,- DM vor Ort erhoben.

PROGRAMMKOMITEE:

R. Dumke, Universität Magdeburg (Vorsitz)
H. Herting, DeTeCSM GmbH Darmstadt
R. Hopfer, Hochschule für Technik und Wirtschaft Dresden
F. Lehmann, Universität der Bundeswehr München
C. Rautenstrauch, Universität Magdeburg (Vorsitz)
A. Schmietendorf, T-Nova GmbH Berlin
A. Scholz, Universität Magdeburg
W. Schröder, DeTeCSM Magdeburg
F. Victor, Fachhochschule Köln

TERMINE:

07. April 2000: Einreichung von Beiträgen
21. April 2000: Benachrichtigung über Annahme/Ablehnung
21. April 2000: Versand des endgültigen Workshopprogramms
05. Mai 2000: Anmeldeschluß zum Workshop
15. Mai 2000: Workshop in Darmstadt

C A L L F O R P A P E R S

10th International Workshop on Software Measurement

of the German Interest Group on Software Measurement and Evaluation (GI-FG 2.1.10), the Canadian Interest Group on Metrics (C.I.M.), and the Common Software Measurement International Consortium (COSMIC)

October 4-6, 2000
Berlin, Germany

SCOPE

Software measurement is one of the key technologies to control or to manage the software development process. The applicability of metrics, the efficiency of metrics programs in industry and the theoretical foundations have been subject of recent research to evaluate and improve modern software development areas such as object-orientation, component-based development, multimedia systems design, reliable telecommunication systems etc. Our recent workshops have been attentive to these concerns. Research initiatives were directed initially to the validation of software metrics and their practical use based on critical analysis of the benefits and weaknesses of software measurement programs. But up to now the application of metrics does still involve the risk of failure. Therefore, it is necessary to stimulate further theoretical investigations to improve the engineering foundations in software development and measurement. We are looking for papers in the area of software metrics and software measurement from but not limited to the following areas:

- Experience reports and controlled experiments
- Lessons learned from establishing a measurement program in industry (could be successes or failures with post mortems)
- Theoretical background and further practical applications of the Function Point Method
- Metrication of new OO languages, components or methods such as Java, JavaBeans or UML
- Metrics data bases, repositories and experience factories
- Software Measurement for Web-based applications
- Development and use of measurement tools
- Theory of measurement and its practical implications
- Methods and systems evaluation of different kinds of embedded, knowledge-based and communication systems.

New ideas, original approaches and fundamental concepts which can and should be discussed are welcome.

PROGRAM COMMITTEE

Alain Abran, University of Quebec, Montreal, Canada
Manfred Bundschuh, DASMA, Germany
Jean-Marc Desharnais, CIM Montreal, Canada
Reiner Dumke, University of Magdeburg, Germany
Claus Lewerentz, TU Cottbus, Germany
Rini van Solingen, IESE Kaiserslautern, Germany
Andreas Schmietendorf, T-Nova Berlin, Germany
Harry Sneed, SES Munich/Budapest, Hungary

SUBMISSIONS

Authors should send full papers (max 12 pages) for 30 minutes presentastion (including discussion) by mail, fax or email by June 16, 2000 to

Alain Abran
University of Quebec
Dept. Of Computer Science
C.P.8888, Succ. Centre-Ville
Montreal (Quebec), Canada H3C 3P8
Tel.: +1-514-987-3000
Fax: +1-514-987-8477
Email: abran.alain@uqam.ca

or to

Reiner Dumke
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Postfach 4120
D-39016 Magdeburg, Germany
Tel.: +49-391-6718664
Fax: +49-391-6712810
Email: dumke@ivs.cs.uni-magdeburg.de

WORKSHOP TIMETABLE:

Submission deadline: June 16, 2000
Notification of acceptance: July 21, 2000
Copies of presentation slides to prepare a proceedings draft: August 31, 2000
Full paper for publishing in a metrics book: workshop date
Workshop date: October 4-6, 2000

NEWS

For the latest news about the Workshop please see the following Web site:

http://ivs.cs.uni-magdeburg.de/sw-eng/us/IWSM2000/

International Workshop on Software Measurements
IWSM'99
Lac Superieur, Québec, Canada, September 8 - 10, 1999
Abstracts

Tool Description

APPLICATION DE LA METHODE FFP A PARTIR D'UNE SPECIFICATION SELON LA NOTATION UML : COMPTE RENDU DES PREMIERS ESSAIS D'APPLICATION ET QUESTIONS

Valéry Bévo, Ghislain Lévesque et Alain Abran
Université du Québec à Montréal
Département d'informatique
C.P. 8888, Succ. Centre-Ville
Montréal, Québec, Canada
H3C 3P9
[valery.bevo@lrgl.uqam.ca, levesque.ghislain@uqam.ca, abran.alain@uqam.ca]

RÉSUMÉ

La mesure apparaît aujourd'hui comme un outil nécessaire pour les analyses de qualité et de productivité associées au développement et à la maintenance de logiciel. Plusieurs méthodes de mesure ont vu le jour : F.P.A. (Function Points Analysis), MarkII, F.F.P. (Full Function Points), et bien d'autres.

La méthode F.F.P. s'avère efficace pour la mesure de la taille fonctionnelle des logiciels aussi bien temps-réel qu'embarqués, de gestion ou système.

Sur un tout autre plan, UML(Unified Modelling Language) est une notation de plus en plus utilisée dans le domaine de la spécification et de la conception de logiciel. Elle fournit un vocabulaire et des règles pour représenter les différents modèles permettant de comprendre (de visualiser) un système. Elle permet de représenter un système à toutes les étapes de sa réalisation.

Quel lien faisons-nous entre la notation UML et la méthode FFP ?

La phase de « mapping » du processus de mesure FFP apparaît comme une phase critique. En effet, la seconde phase du processus (la phase de mesure) en dépend : Plus le modèle FFP du logiciel (obtenu à l'issue de la phase de « mapping ») est complet, plus la mesure de la taille fonctionnelle du logiciel (obtenue à l'issue de la phase de mesure) est significative.

Le succès de la phase de « mapping » dépend pour beaucoup de la qualité des documents de spécification du logiciel fournis en entrée de la phase. UML s'imposant peu à peu comme une notation standard pour les méthodes d'analyse et de conception orientées-objet, de plus en plus de documents de spécifications de logiciel seront basés sur cette notation. Par conséquent, il nous a semblé opportun d'examiner la possibilité, à partir d'un document de spécification en UML d'un logiciel, de déterminer la taille fonctionnelle du logiciel à l'aide de la méthode de mesure FFP.

A défaut de pouvoir déjà proposer une solution à ce problème, nous identifions dans le présent document les questions de fond qui se posent pour sa résolution. Nous commençons par établir un rapprochement entre certains concepts FFP et UML. Sur la base de ce rapprochement, nous identifions à partir d'une étude de cas (mesure de la taille fonctionnelle d'un logiciel de contrôle d'accès à un bâtiment[1] à partir de son document de spécification en UML), les principales difficultés à surmonter pour aboutir à une solution acceptable, sans nécessairement préciser comment les surmonter. Notre proposition vise à susciter le débat autour de la question, lequel débat devrait permettre de s'orienter vers une solution acceptable.

[1] Modélisation objet avec UML, Pierre-Alain Muller, Chapitre V

3

Tool Description

X-RAY: A MULTI-LANGUAGE, INDUSTRIAL STRENGTH TOOL

Sue Black and David Wigg
Centre for Systems and Software Engineering
South Bank University
London SE1 0AA
UK
e-mail: {blackse, wiggjd} @sbu.ac.uk

ABSTRACT
The software industry needs software measurement tools which can be used to compute measures across several platforms and languages to provide flexibility and usability. The X-Ray program structure analyser goes a long way towards delivering these goals by providing the raw data for analysis. X-Ray is a general purpose program for the structural analysis of programs which have been written in a wide variety of procedural languages. In the process of structural analysis it can produce specific information for software measurement including size and complexity measures. X-Ray output is used in the production of flow-graphs for the QUALMS structure analysis system and has been used to provide comprehensive data for the measurement of industrial C++ code.
Keywords: structural analysis, C++, legacy systems.

LIME: A THREE-DIMENSIONAL MEASUREMENT MODEL FOR LIFE CYCLE PROJECT MANAGEMENT

Luigi Buglione
European Software Institute
SPI Measurement Product Line
Parque Tecnológico de Zamudio #204
E-48170 Vizcaya, Spain
E-mail: luigi.buglione@esi.es
Tel: (34) 94.420.9519
Fax: (34) 94.420.9420

Alain Abran
Software Engineering Management
Research Laboratory
Université du Québec à Montréal
C.P. 8888, Succ. Centre-Ville
Montréal, Québec, Canada
E-mail: abran.alain@uqam.ca
Tel: +1 (514) 987-3000 (8900)
Fax: +1 (514) 987-8477

INDEX – 1. Introduction – 2. Software Life Cycle Models – 3. Management of Quality during SLC – 4. The QEST model – 5. The LIME model – 6. Conclusions & Prospects – References

ABSTRACT
Organizational performance models are usually based on accounting systems, and therefore take into account mostly the economic-financial viewpoint, or the *tangible asset* part, of it using performance management terminology. In the IT field, the Earned Value model has been promoted to be present project performance during the project life cycle. However, these types of models oversimplify performance representation with a single performance index, while in reality multiple viewpoints must be managed simultaneously for proper performance management.This work shows how an *open* three-dimensional measurement model of software project performance functions. Called **LIME** (**LI**fecycle **ME**asurement), it extends the structure of a previous model to a dynamic context I applies to software production during all SLC phases, which are classified following a generic 6-step and scheme waterfall standard. A quantitative and qualitative analysis of the project is effected considering the three distinctive but connected areas of interest, each of them represent has a dimension of performance:

- **economic dimension**, from the *managers'* viewpoint, with a particular attention to cost and schedule drivers;
- **social dimension**, from the *users'* viewpoint, with particular attention to the quality-in-use drivers;
- **technical dimension**, from the *developers'* viewpoint, with particular attention to technical quality, which has a different impact during each SLC phase.

Keywords: Performance Measurement, Software Product Quality, Metrics, Function Point Analysis, ISO/IEC 9126, GQM approach, SLC, QEST model.

«REVENUE CANADA'S EXTENDED METRICS PLAN» OR «METRICS WITH MUSCLE»

Valérie Burton et Linda Albert
Revenue Canada, Customs, Excise and Taxation – Federal Department

ABSTRACT
Our presentation is delivered using the nine commandments of innovation. We need to be innovative because what we need to do is not an existing process. We explain.

- our passion
- understanding of where we currently are to achieving this passion,
- how we evolved to the point of being able to realize this passion,
- how we identified and analyzed the gap between ourselves and a «world-class» program,
- how the Extended Metrics Plan was created,
  - the barriers to it being a success,
  - how to go about achieving buy-in,
  - how we intend to complete it by defining our market and assessing it's impact.

We've discuss the defined phases of the plan plus the

- inputs,
- processes, and
- outputs for each phase.

In short, a «how to» manual with measurable deliverables and a timetable.

SOFTWARE LAYERS AND MEASUREMENT

Jean-Marc Desharnais, *Software Engineering Laboratory in Applied Metrics*
Denis St-Pierre, *DSA Consulting Inc.*
Serge Oligny, *Laboratoire de recherche en gestion des logiciels*
Alain Abran, *Laboratoire de recherche en gestion des logiciels*

ABSTRACT
Systems rarely run alone. They are usually part of a complex system of software layers (e.g. database managers, network drivers, operation systems and device drivers). Software layers constitute a specific way of grouping functionalities on a level of abstraction.
When measuring the functionality of a system, practitioners usually consider one type of layer: user application, or the highest-level layer. They consider the other layers as technical. This approach might work with Management Information Systems, where there is often no

business need to consider layers other than the highest-level one. This is because the other layers are usually already developed (e.g. Windows, UNIX, printer drivers). However, this is often not the case for real-time and embedded systems. Embedded system development projects involve developing or modifying operating systems, drivers and user applications as well. Not considering software layers can result in misleading measurements, as measuring only the highest-level layer may lead to misrepresentation of the size of a project or application.
This paper covers the definition of software layers and how to identify them, and by extension the identification of peer systems: systems residing on the same layer.

DECISIONS AND JUSTIFICATIONS IN THE CONTEXT OF INDUSTRIAL-LEVEL SOFTWARE ENGINEERING

Françis Dion
ESI Software inc.
E-mail: FDion@esisoft.com
Phone: (514) 745-3311 ext. 283
Fax: (514) 745-3312

Alain Abran
Software Engineering Management Research Laboratory
Université du Québec à Montréal
C.P. 8888, Succ. Centre-Ville
Montréal, Québec, Canada
E-mail: abran.alain@uqam.ca
Phone: +1 (514) 987-3000 (8900)
Fax: +1 (514) 987-8477

ABSTRACT
Decision-making is a difficult task per se. This inherent difficulty is exacerbated by the complexity and fast pace of the changes that characterize software engineering. Critical decisions impacting the success of a project or even an entire organization must be made quickly based on information that is either limited to the point of being insufficient or so abundant that it is virtually unmanageable. Either way, the information is more often than not of questionable quality.
This paper proposes an evolutionary framework to support efficient and justifiable decision-making throughout the implementation phase. This approach covers the necessity to make decisions quickly without complete, reliable information, as well as integrate new data as it becomes available.

A FRAMEWORK FOR SOFTWARE MEASUREMENT EVALUATION

Reiner R. Dumke
Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
Postfach 4120, D-39016 Magdeburg, Germany
dumke@ivs.cs.uni-magdeburg.de

ABSTRACT
This paper is written to motivate the (world-wide) software metrics community to consider the possibilities of standards for software measurement classification and implementation as a persistent control element in the software development and maintenance. In practice we can establish that we have an unsatisfactory situation after an ISO 9000 certification or Capability Maturity Model (CMM) evaluation. These evaluations do not include a classification of the software measurement 'level' itself. Hence, we need measures or metrics to characterize the software measurement process.

Therefore, we define a measurement framework to evaluate the software development and maintenance in this manner that we consider all the aspects to obtain a level in the sense of the CMM that is a 'true level four'.

Keywords: Software measurement, measurement evaluation, OO metrics, metrics applications

A Generalized Structure for Function Point Analysis

Thomas Fetcke
Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
Postfach 4120, D-39016 Magdeburg, Germany

Abstract

Since its first publication by Albrecht, Function Point Analysis has been revised and modified several times. Today, a number of variants are in use, which differ in their respective views on functional size.

Function Point Analysis relies implicitly on a model of software. We propose the Function Point Structure as a formalization of the software model of Function Point Analysis. The Function Point count is then defined as a function on the Function Point Structure. Function Point variants differ in their abstract models of software as well as in their measure functions. Therefore, different formalizations of the Function Point Structure are required for each variant. We present here a generalized Function Point Structure for several data oriented variants of Function Point Structure, we can analyze the empirical assumptions made by the FPA variants and the implications on the prediction of other variables. We can also study the differences between the views and assumptions of the variants.

A Function Point Counting Rules Project at Galorath Inc.

Lee Fischman
Lee@galorath.com

Abstract

A function point counting rules project at Galorath Inc. Has aimed at refining and synthesizing function point counting rules, and discerning the patterns that emerge. Researching the logical basis of the counting rules has reduced the number of rules required, simplified them, and clarified common function point counting pitfalls. A rule-based rationale for an "internal function" also has been developed. The counting rules have been synthesized so that a simple flowchart interconnects all rules into a common decision tree, with children representing specific function point outcomes.

The work has so far led to the development of a hypertext tool known as the Function Point Wizard, which can be used to rapidly train people how to count; it also provide clarification for experienced counters. Over the past year, several hundred downloads of the tool have occurred from Galorath's web site.

Requirements and Design of a Metrics Database for Industrial Use

Erik Foltin, *OvG Universität Magdeburg*
foltin@ivs.cs.uni-magdeburg.de
Andreas Schmietendorf, *Deutsche Telekom Berlin*
schmietendorf@05.bln01.telekom400.dpb.de
Reiner Dumke, *OvG Universität Magdeburg*
dumke@ivs.cs.uni-magdeburg.de

ABSTRACT
There exist a lot of methodology hints, software inspection- and review- handbooks, and metrics tools to support the application of software measurement throughout the software lifecycle. There is still, however, a widespread lack of confidence in the interpretation of metrics and their values, in concentration on use of only a few metrics and in the processing of large sets of measured values. The analysis of the different evaluation, counting and measurement results should be supported by a data base technique to keep control of the software development process. This paper describes the different sources of metrics data and their effective handling. It concludes with a discussion of the experiences with a prototype of a metrics database in an industrial setting.
Keywords: software quality assurance, software process controlling, software metrics, metrics data base

COUNTING FUNCTION POINTS FROM B SPECIFICATIONS

H. Diab, M. Frappier, R. St-Denis
Département de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke, Québec, Canada J1K 2R1
Email: {hassan.diab, marc.frappier, richard.st-denis}@dmi.usherb.ca
D. Déry
Groupe CGI Inc.
1800, McGill College
Montréal, Québec, Canada H3A 3J6
Email: david.dery@cgi.ca

ABSTRACT
Function points is one of the prominent methods which has gained a considerable popularity in the industry to measure functional software size. It has been proposed by Albrecht, and it was later refined by some organizations like the International Function Points Users Group (IFPUG) in order to facilitate its industrial use. This paper proposes a formalization of the IFPUG definition of function points using the formal specification language B. The goals of this formalization are twofold. The first one is to provide an objective definition of function points, which should reduce variance in function point counts due to rater interpretation. The second one is to automate function point counts for B specifications, which should reduce measurement costs.

ESTABLISHING SOFTWARE METRIC THRESHOLDS

Vern A. French
PRIOR Data Sciences Ltd.
Kanata, Ontario, Canada

ABSTRACT
This paper describes a method for establishing software metric thresholds that narrows their focus to the truly problematic code. The method allows conventional software metrics to be used on languages based on fundamentally different development paradigms and methodologies. It also allows metric thresholds to be adapted to compensate for unique project characteristics and conditions.

AUTOMATIC FUNCTION POINT COUNTING USING STATIC AND DYNAMIC CODE ANALYSIS

Keith Paton
350 Pine Avenue
Saint Lambert (Quebec)
Canada
J4P 2N8
Tel (450) 671-1969
Fax (450) 465-9386
internet paton@total.net

SUMMARY

We define an intermediate representation of a program P as a data flow graph DF(P) and shown that this representation allows us to express the program as a quadruple Q={F,T,r,w} useful in function point analysis.

We show that we can derive DF(P) by program slicing, a form of static code analysis. Starting with one given output file, A say, we derive the smallest program A(P) that mimics P in its writing to A. When we repeat this process for all output files,(A,B,C,D say) we obtain a set of programs A(P), B(P), C(P) and D(P) in which any two are disjoint or identical. The number of unique such programs is the number of transactions and straightforward analysis yields DF(P).

We shown that we can also derive DF(P) by program tracing, a form of dynamic code analysis. In this case, we can modify the program P being studied into a second program P' such that P' has the same behavior as P and P' generates a trace showing what it is doing. From the trace, we can automatically derive the intermediate representation DF(P). The modification of P to P' can be carried out automatically by methods of static code analysis now under development.

UNIVERSALITY – A NEED FOR A NEW SOFTWARE METRIC

Peter Kokol, Vili Podgorelec, Milan Zorman
University of Maribor, FERI, Laboratory for System Design
Smetanova 17, 2000 Maribor, Slovenia
E-mail: kokol@uni-mb.si

INTRODUCTION: THE NEED FOR UNIVERSALITY

Although the software metrics are becoming more and more recognised in software engineering, the traditional metrics still have many deficiencies:

They are language dependent. We have to use a different metric to assess a program written in a higher level programming language (even worse we have to use a different metric or at least a different tool for each programming language), another metric for a program written in an assembler language and again some other metric to asses the complexity of object or executable code.

1. *A program or an information system can be represented in many forms*: requirements, specification, documentation, user interfaces, etc. and all that representations can be manifested in very different appearances: written text, graphical, symbolic, formal

languages, natural languages, etc. – a different metric for each representation is needed again.

To overcome above deficiencies we need an universal metric with well defined critical values, and to construct such a metric we employed the ideas found in the new formed scientific discipline called complexity.

MULTIPLE VIEWPOINTS IN FUNCTIONAL SIZE MEASUREMENT

Christopher Lokan
School of Computer Science
University of New South Wales
Australian Defence Force Academy
Canberra ACT 2600, Australia
+61 2 6268 8060
c-lokan@adfa.edu.au

Alain Abran
Université du Québec à Montréal
Département d'informatique
C.P. 8888, Succ. Centre-ville
Montréal (Québec), Canada H3C 3P8
+1 514 987 3000 (8900)
abran.alain@uqam.ca

ABSTRACT

Although there is broad agreement on the sorts of things to take into account when measuring functional size, there is a variety of opinion about how to do it. This is partly because several different views of functionality are addressed in functional size measurement. Some are better understood than others. In particular, the "general systems characteristics" (GSC's) and "value adjustment factor" (VAF) are poorly understood. Our aim is to provide a foundation for research that may improve this aspect of functional size measurement.

A survey of the evolution and state of practice of the GSC's and VAF leads us to identify various aspects of software that are important in functional size measurement. We relate these aspects of software to different views of functionality. A spectrum of viewpoints is seen, with core functionality at one end, effort estimation at the other, and different user viewpoints in between. By noting how the GSC's and VAF contribute to these viewpoints, we see how value may be gained from them, and we identify directions for future research.

ESTABLISHING SOFTWARE SIZE USING THE PAIRED COMPARISONS METHOD

Eduardo Miranda
eduardo.miranda@lmc.ericsson.se

INTRODUCTION

In our everyday life, we rely on measurement scales and measurement instruments to make all kind of decisions: We listen to the radio to learn about the temperature before deciding how to dress, we look at the Dow Jones index before making an investment, and we calculate the distance separating two cities before starting a trip. But, what do we do when we need to establish the software size at the beginning of a project, before a detailed specification or draft design exists?

Lines of code and function points are good examples of size metrics, but the counting process requires effort and information, which in too many projects, is only available after the project budget and schedule have already been decided. Furthermore, product managers and other stakeholders with non-software background, simply refuse to accept measures that are foreign to them.

To solve the problem of measuring in the absence of a unique and accepted measurement scale, or in cases were a measurement instrument does not exist, the social sciences have

adopted the method of paired comparisons for establishing the relative merit of an entity with respect to others. Entities in the social sciences take the form of attitudes, preferences, brand recognition, etc.
The same approach could be used to size software. Although the idea is not new, it has received very little attention in the literature. Earlier attempts include Target Software's Software Sizing Method [1], and more recently a paper by Focal Point AB [2] where an instance of the method, called the Analytic Hierarchy Process, is used to prioritize requirements relative to their cost.
The idea behind the paired comparisons method, is to estimate the size of *n* entities, be these tasks, requirements, use cases, modules, features or objects, based on their relative largeness as judged by one or more experts.
This article explains the advantages of the approach, the selection of scales, the computational methods and the necessary tool support.

SOFTWARE OUTSOURCING CONTRACTS: AN ECONOMIC ANALYSIS BASED ON AGENCY THEORY

Luis Molinié & Alain Abran
Software Engineering Management Research Laboratory
Université du Québec à Montréal
C.P. 8888, Succ. Centre-Ville
Montréal, Québec, Canada
E-mail: d327634@er.uqam.ca abran.alain@uqam.ca
Tel: +1 (514) 987-0376
Fax: +1 (514) 987-8477

ABSTRACT
This paper presents an analysis of contractual outsourcing agreements in the field of Information Technology based on the postulates of the Agency Theory. This analysis reveals that the design of many outsourcing agreements, referred to as procurement contracts, is incomplete from an economic perspective. It is postulated that this degree of contractual incompleteness is the result of a trade-off between the benefits of mitigating the ex-post opportunism of agents and the costs of additional resources expended in ex-ante design. The magnitude of these opposing forces can be predicted based on the characteristics of the suppliers and the software services.
From this postulate, as well as from previous findings in the literature on manufacturing procurements, this paper suggests a model which links the degree of contractual completeness with some variables related to the potential opportunism of suppliers and the uncertainty surrounding software services. A subsequent research phase will test this model in software outsourcing environments.

CONCEPTION AND EXPERIENCE OF METRICS-BASED SOFTWARE REUSE IN PRACTICE

Andreas Schmietendorf, *Deutsche Telekom AG, Entwicklungszentrum Berlin*
Evgeni Dimitrov, *Deutsche Telekom AG, Entwicklungszentrum Berlin*
Reiner Dumke, *Otto-von-Guericke-Universität Magdeburg*
Erik Foltin, *Otto-von-Guericke-Universität Magdeburg*
Michael Wipprecht, *Deutsche Telekom AG, Entwicklungszentrum Berlin*

ABSTRACT
There are already a number of studies and »success stories" about practical applications related to software reuse. For the most part however, the actual benefits of reuse, particularly for concrete technologies, are difficult to verify. The SW-WiVe project performed by Deutsche Telekom in collaboration with the Otto-von-Guericke University provides a detailed analysis and offers strategies for software reuse within industrial software development that can be subjected to critical evaluation. Traditional evaluation approaches, such as reuse metrics, were critically studied and necessary processes for continuous reuse were developed for this purpose. In a further step, currently available, valid reuse metrics for the software development process were classified and lacking metrics-based evaluation approaches were identified. This paper focuses on a description of the project's metrics-oriented terms of reference.

USING UML ELEMENTS TO ESTIMATE FEATURE POINTS*

Richard D. Stutzke
Science Applications International Corp.
6725 Odyssey Drive, Huntsville, AL 35806-3301
(256) 971-6224 (office) / (256) 971-6550 (facsimile)
Richard.D.Stutzke@cpmx.saic.com

ABSTRACT
Why do we size software? For many reasons. To estimate development effort. To estimate memory requirements. To estimate processing or execution speed. To estimate the value of software assets. This paper considers size measures that help us predict the effort needed to design, build and test computer software using object-oriented methods. We estimate the size by combining estimates for the three aspects of software: data, control behavior, and function, as identified by [DeMarco, 1982]. The Unified Modeling Language (UML) is emerging as an industry standard for recording the results of object-oriented analysis and design. We focus specifically on the use of the products of the UML as early reliable measures of software size. We identified a subset suitable for early estimation of effort by examining the development process. We examined the elements of UML notation, and the activities of the Object-Oriented Systems Analysis (OOSA) method. This method is similar to Object Management Technique (OMT) which has been absorbed into UML. The Class Diagram (essentially an Entity Relationship Diagram) and the Event List or, better, the Statechart (a State Transition Diagram) seem to suffice to determine most of the development effort. The only activity we think is not covered is the development of complex operations (algorithms, methods). We recommend an approach to help the estimator identify these and estimate the additional effort. We propose a way to use these UML elements to estimate the size in Unadjusted Feature Points. We conclude by identifying some questions that need to be addressed in the future.

COSMIC - AIMS, DESIGN PRINCIPLES AND PROGRESS

Charles Symons, Alain Abran
P. Morris, P. Fagg, et al.

* This paper is an abridged version of [Stutzke, 1998]. We have revised the equations slightly and added some new references.

Tool Description

ABSTRACT
COSMIC, the Common Software Measurement International Consortium with participation from leading software measurement experts from Australia, Europe and North America aims to develop a new generation of measures of functional size for use in performance measurement and estimating in software activities. The measures will draw upon the best features of existing IFPUG, NESMA, MkII and Full Function Point methods, the emerging ISO standards, and new ideas, to achieve higher levels of accuracy and much wider applicability, specifically across both business and real-time software, than existing functional sizing methods.
This presentation will briefly review the origins and needs for the COSMIC approach, then discuss its design principles and report on the project's current organization and status.

A FRAMEWORK FOR AUTOMATIC FUNCTION POINT COUNTING FROM SOURCE CODE

Vinh T. Ho and Alain Abran
Software Engineering Management Research Laboratory
Université du Québec à Montréal (Canada)
vho@lrgl.uqam.ca abran.alain@uqam.ca

ABSTRACT
The paper proposes a general framework to build a model for automatic Function Point Analysis (FPA) from the source code of COBOL system using program slicing technique. The COBOL system source code is scanned by the model to produce Function Point counts. The application's source files are used to define the application's boundary for the count. The model takes into account the structure of the COBOL language to identify physical files and transactions. Reserved words as FDs, file input/output statements (READ and WRITE) and user interface and data manipulation statements (ACCEPT, DISPLAY and MOVE) are used as basic information for program slicing technique to identify candidate physical files and transactions. Some heuristic rules will be proposed in order to map candidate physical files and transactions into candidate logical files and transactions. These candidate files and transactions are then assessed with regards to the IFPUG' identifying rules in order to identify data function types and transactional function types to be counted. The proposed framework helps to build models for automating Function Point Analysis from source code in compliance with the IFPUG Counting Practices Manual.

EVALUATING MEASUREMENT METHODS

Leonard White, *Manger*
The Everest Group
North American Operations

ABSTRACT
The number of methods for measuring applications has grown to the point that it is becoming difficult to choose the appropriate measurement method. This presentation will present methods for comparing different measurement methods. The presentation should be of interest to intermediate and advanced measurement practitioners.
The topics that covered in this presentation are:
1. Determining what is measured, the product or the process that produces the product

2. Evaluating the attributes of the product or process
3. Evaluating the precision required to quantify the attribute of the product or process
4. Evaluating the combination of measurement to determine the correct metric ratio
5. Evaluating the measurement of functionality

Examples from Function Points, time, cost and quality measurements will be used to illustrate the points.

The people attending this presentation will learn how to compare different measurement methods. The information from this presentation will enable the creation of more robust and scalable measurement methodologies.

VALIDATION OF MEASURES AND PREDICTION MODELS

Horst Zuse
Technische Universität Berlin
Department of Computer Science
Franklinstraße 28/29, FR 5-3
10587 Berlin / Germany
E-mail: zuse@cs.tu-berlin.de
Phone: +49-30-314-24788 / Fax: +49-30-314-73489
Internet: http://www.cs.tu-berlin.de/~zuse

ABSTRACT

Validation and prediction are closely related terms. Mostly, measures are validated in order to predict an external variable. Such external variables can be costs of maintenance, time to repair a module, etc. Validation of software measures is a very important task, but not an easy one. Mostly, correlation coefficients or regression analysis for the validation of software measures are used. In this paper a set of fundamental properties of prediction models is considered. The result is, that only a certain set of measures are appropriate for a prediction model. The only one possible prediction function then is the formula of the basic COCOMO-Model.

Keywords: Prediction, Prediction models, independence conditions, measurement, software measurement, extensive structure, validation, COCOMO Model.

CALCUL DES POINTS DE FONCTION A PARTIR DU DIAGRAMME DES CAS D'UTILISATION DE LA NOTATION UML.

S. Labyad, M. Frappier et R. St-Denis
Université de Sherbrooke
Département de mathématiques et d'informatique
Faculté des Sciences
Sherbrooke, Québec, CANADA J1K 2R1
E-mail: {labyad, frappier, stdenis} @dmi.usherb.ca

ABSTRACT

L'étude préliminaire est constituée des actions sur la prise de connaissance du cahier des charges ou de l'expression du besoin. Elle ne porte que sur les aspects externes du système à réaliser. C'est la recherche du "quoi" faire par rapport au futur "comment" faire de la conception.

Les éléments de cette phase sont constitués par les diagrammes des cas d'utilisation du système et des descriptions textuelles. Les exigences et les contraintes à satisfaire sont

éventuellement reprises ici (si cela n'a pas déjà été fait au niveau du cahier des charges ou des annexes techniques liées au contrat). Le dossier d'étude préliminaire peut comporter deux parties:

- l'analyse des spécifications du système qui porte sur une décomposition fonctionnelle des fonctions à réaliser,
- la conception de l'architecture du système qui comporte une décomposition en sous-système et les descriptions des interfaces fonctionnelles entre sous-systèmes.

Dans le cas d'une refonte d'un système d'information, une partie associée à l'analyse du système existant peut être ajoutée. Un dossier de spécification du système (cas d'utilisation du système) est fourni.

Un cas d'utilisation, concept initialement introduit par Jacobson, est défini comme une manière spécifique d'utiliser le système en faisant appel à une partie de sa fonctionnalité ou encore comme une suite particulière de transactions liées les unes aux autres, et dont l'origine est un dialogue entre un agent et le système. Il y a plusieurs travaux qui portent sur les cas d'utilisation concernant la modélisation, la formalisation et la documentation [6, 7, 8, 9, 10, 11]. Peu de travaux existent sur le calcul des points de fonction à partir de diagramme des cas d'utilisation [12, 13].

Dans cet article, nous nous intéressons au calcul des points de fonction à partir des diagrammes de cas d'utilisation de la notation UML [14]. L'originalité de notre travail de recherche réside dans l'exploitation de tous les concepts utilisés dans un diagramme de cas d'utilisation : acteur (acteur primaire et acteur secondaire), cas d'utilisation, relations entre les cas d'utilisation (relation d'utilisation, relation d'extension et relation de communication) et la documentation du cas d'utilisation.

Un cas d'utilisation nécessite, pour être compris, une documentation plus détaillée. Cette documentation peut être donnée sous forme d'un diagramme de séquences. Les diagrammes de séquences permettent de spécifier des séquences de messages échangés entre des objets. Ces diagrammes, lorsqu'ils sont utilisés pour décrire un cas d'utilisation, ne comportent que deux catégories d'objets : les acteurs externes et des composants du système qui interagissent directement avec les acteurs. Seuls les interactions vers ou provenant d'un acteur sont considérées ici. Les messages provoqués par ces interactions n'apparaîtront qu'ultérieurement lors de la description des scénarios internes au système. Aucune classe n'est encore évoquée explicitement ici. Un diagramme des séquences associé à un cas d'utilisation permet donc de spécifier uniquement les échanges de données ou d'événements entre un utilisateur du système et le système lui-même.

Les composants relatifs aux données et les composants relatifs aux transactions sont parmi les concepts clés des points de fonction. Pour identifier les composants relatifs aux données (GDI et GDE) nous avons examiné les acteurs (acteurs primaires et acteurs secondaires) afin de décider lesquels représentent un des concepts des points de fonction. Selon Jacobson [5], les acteurs primaires vont directement utiliser le système. Chacun de ces acteurs effectuera une ou plusieurs des tâches principales du système. Les acteurs secondaires (intermédiaires) supervisent et maintiennent le système. L'acteur secondaire n'existe que pour permettre à l'acteur primaire d'utiliser le système. Lors du passage du modèle des cas d'utilisation au modèle objet, l'acteur primaire va devenir une classe. Donc, nous pouvons dire qu'un acteur primaire est un GDI et un acteur secondaire ne l'est pas sauf si nous voulons conserver des instances sur cet acteur. De plus, nous avons exploité la documentation (les diagrammes de séquences) des cas d'utilisation pour mieux identifier d'autres groupes de données (GDI et GDE). Nous avons examiné chaque événement et l'ensemble des informations (DE) échangées entre les acteurs, le système (ou composant du système) et les autres systèmes afin d'identifier les GDIs et les GDEs. Pour identifier les composants relatifs aux transactions, nous avons

examiné la documentation (les diagrammes de séquences) des cas d'utilisation. Selon [12, 13], un cas d'utilisation peut contenir une ou plusieurs transactions. Toutefois, on n'y indique pas comment on identifie les entrées, les sorties et les interrogations de manière explicite. La pertinence de notre travail de recherche réside dans la façon précise d'identifier les GDIs, les GDEs, les entrées, les sorties et les interrogations à partir de l'ensemble des concepts du diagramme des cas d'utilisation et leur documentation. Enfin, nous proposons une ébauche d'un algorithme, que nous avons expérimenté sur un exemple concret afin de calculer la taille d'un logiciel en points de fonction à partir du diagramme des cas d'utilisation de la notation UML [14, 15, 16].

Tool Description

THE WORKSHOP OF THE GI-ARBEITSKREIS SOFTWAREMETRIKEN (FG 2.1.9)

Our annual workshop of the GI-Arbeitskreis Softwaremetriken would be held in Regensburg in September 30 - October 1, 1999.

The presented papers was the following:

- M. Jacobsen-Rey (Reasoning Software GmbH, Waldems-Esch):
  Automated Software Inspection - Attaining New Levels of Software Quality
- T. Fetcke (University of Magdeburg):
  A Generalized Structure for Function Point Analysis
- E. Foltin (University of Magdeburg):
  Entwicklung einer industriell nutzbaren Metriken-Datenbank
- C. Lewerentz (Technical University of Cottbus):
  Quality - Metrics - Numbers - Consequences - Lessons Learned
- R. Dumke (University of Magdeburg):
  Anwendungserfahrungen mit einem allgemeinen Measurement Framework
- A. Schmietendorf (EZ Telekom, Berlin):
  Konzeption und erste Erfahrungen einer metrikenbasierten Software-Wiederverwendung
- P. Mandl-Striegnitz (University of Stuttgart):
  Untersuchung eines neuen Ansatzes zur Projektmanagement-Ausbildung
- H. Windpassinger (Verilog, Munich):
  Möglichkeiten der metrik-basierten Modellierung und Auswertung von Qualitätsvorgaben mit dem Werkzeug LOGISCOPE
- F. Simon (Technical University of Cottbus):
  Semiautomatische, kohäsionsbasierte Subsystembildung
- U. Schweikl (Siemens AT, Regensburg):
  Applicability of Full Function Points at Siemens AT
- H. Sneed (SES Munich/Budapest):
  Testmetriken für objektorientierte Bankanwendungen
- C. Ebert (Alcatel, Antwerp, Belgium):
  Process Change Management in large Organizations
- A. Mittelmann (VOEST-Alpine, Linz):
  Messen von weichen Faktoren - Ein Erfahrungsbericht
- M. Rickheit (QA Systems, Stuttgart):
  Erste Schritte bei der Einführung von Software-Metriken

The paper are sumarized in a book as

Dumke/Lehner: Software-Metriken - Entwicklungen, Werkzeuge und Anwendungserfahrungen. DUV, Wiesbaden, 2000.

This book should be available in the February 2000.

GARTNERGROUP SYMPOSIUM/ITXPO99 - THE FUTURE OF IT

Andreas Schmietendorf
Otto-von-Guericke-University of Magdeburg, Faculty of Computer Science

ABSTRACT
This year the annual European conference of theGartnerGroup was held between November 12 and 16, 1999 in Cannes, France. During this conference parallel sessions covered diverse topics from the context of the information-technology IT, including applications-development, the business of information system as well as knowledge management. Within this article we will focus on application development, the extensive use of metrics within the Gartner-analyses as well as presentations concerned directly with the introduction and use of software-metrics programs.

1 Introduction

The rapid development of the information-technology requires the IT-management to deal with extensive technical and economic information. Such information is often provided by independent analysts like the GartnerGroup described in this report.

The aim of the GartnerGroup is to support organizations to find the right decisions in the area of information technology. Therefore the GartnerGroup provides wide variety of analysis. Offered analyses can be aimed at the selection of new technologies and/or tools to be introduced into the client's business process; finding the corresponding market potentials, the selection of a possible partner as well as for the definition of long-term strategies.

More than 9000 costumers in the field of information technology use analyses provided by Gartner in their decision-making process. The annual conference gives the participants an overview on the state of the art and shows the most important trends for a period of approximately five years.

Such analyses can not replace the necessary competence of an enterprise in the field of information technology. But the analyses offers a preselection of the corresponding subject areas, a faster identification and conversion of the own strategies is therefore possible.

2 Overview about the subjects of the conference

In total there were 41 subject areas covered by 224 presentations the conference participants could select from. The presentations offered a diverse insight into topics like the development and/or the integration and maintenance of information systems, future "call centers technologies" or new technical quality behavior of database management systems.

The conference commenced with a panel discussion about the subject “The social implications of IT”. The omnipresent use of new communication medias was generally in the center of the discussion, focussing on the Internet and the accompanying changes both for the

business world and for the society. Social modifications include for instance new types of virtual universities, the susceptibility from governments and the social life.

In particular for the field of software development, the subject areas (tracks) represented briefly in the following, were from interest. These became thickset at the conference by diverse lectures [Gartner 1999c]:

Application Development and Management: Application development organizations are entering a period of dramatic change, in which roles, responsibilities, technologies, human resources and deliverables are in flux. The presentation of this track showed strategies for the navigation of application development through the next 5 years.

Applications Integration: The reuse of software components up to the re-use of well established information systems as a whole will influence future development of new solutions. Possible strategies, usable methods and tools were represented in this subject area with a focus on the task of integration of available and bought software components(COTS Commercial off-the-shelf).

Internet as E-Business Infrastructure: The use of Internet technology within and between corporations has become a mainstream issue as the Internet is rapidly becoming the primary communication infrastructure for E-Business. The presentations of this track showed major impact that the Internet has in many areas.

E-Business: This subject area represented an emphasis at the conference. The most important evolution in the field "business-to-business" and "business-to-consumer" as well as the redesign of markets following from it were shown. The representations also addressed the necessity of the measurement of successful E-Commerce applications and recording the demands in information technology.

The following topics are restricted to the represented subject areas to the diverse applications of metrics.

3 Gartner analyses and metrics

As far as possible every presentation of the conference followed a common construction. It introduces one of the so-called “key issues” which was deeper explained throughout the course of the presentation. Among these subjects, such questions like assessing the maturity of a technology, identifying market potentials, temporal trends for distribution, strategies- and also action recommendations are represented.

Different elements of visualization are used for the representation of these metrics related subject areas. Diagrams are typical for the temporal behaviour of a feature (e.g. functionality-behaviour, security-aspects, product-maturity), Kiviat-diagrams for the description of different influence criteria.

The GartnerGroup is known for the use of the magic quadrants. The representation of these 4 quadrants within a X/Y-diagram e.g. contains areas like "Niche Players", "Visionaries", "Challengers" and "Leaders". The x-axis describe the “Completeness of vision”, the y-axis

describe the “Ability to execute”. An optimal placement of e.g. a vendor is in the upper quadrant on the right side.

Furthermore the GartnerGroup uses verbal statements which are assigned a probability.

Example: “By 2004, use of SLAs between business units and IS organizations for distributed heterogeneous services will rise from less than 5 percent today to 50 percent (0.8 probability)

From an empirical point of view, the metrics used by the GartnerGroup are ordinal scaled sizes, with a ranking character. Therefore, typically trends are represented; interval- and absolute-scaled values are not used within the GartnerGroup analyses.

4 Lectures with regard to metrics

In general all lectures and/or analyses of the GartnerGroup make extensive use of metrics. Furthermore, the following lectures at this years conference had a direct focus on metrics used during the development, maintenance and management of information systems. The following overview mentions key issues of the presentations. [Gartner 1999c]

- Software process improvement: Application development methodologies, metrics and myth,
  1. What strategies, processes and techniques will assist AD organizations in reducing their exposure to project failures?
  2. Which processes, products and vendors will provide effective methodology solution in support of application delivery?
  3. What strategies, tools and measurement techniques will enable businesses to assess the impact of AD on the organization?

- The evolution of traditional IT measurement techniques,
  1. How will the ability to co-ordinate business and IT change redefine the role of the IS organizations?
  2. How does one measure IT’s role within the enterprise?
  3. What are the key focus areas for IT management?

- Managing IT Quality of Service
  1. What are SLAs (service level agreement) and how are they defined?
  2. How will IS organizations with distributed heterogeneous environment evolve to service-level management?
  3. How will tools evolve to measure against SLAs?

5 Application development and metrics

The presentation „Software Process Improvement - SPI: Applications Development Methodologies, Metrics and Myth” addressed explicitly with the application of metrics in the

context of the application development. For the successful application of metrics, measurement alone is not sufficient. The definition of goals which should be achieved with these metrics is required too. This approach corresponds to the GQM (Goal Question Metrics) paradigm. In the field of the software measurement, this approach is already being employed successfully for a long time. The following general criteria should be considered during the metrics selection in accordance with GartnerGroup. [Gartner 1999a]

1. Metrics must be measurable either by statistics or via observation.
2. They must be linked to the desired performance of application development given the needs of business.
3. They must be comparable to a given organizational goal.
4. Process improvements in software-development requires the attention of diverse criteria.

Gartner proposes that an organization should not simply measure one dimension, but should concentrate on the most important ones. The use of metrics requires a metrics databases. This metrics database system should also include the possibilities for extensive statistical evaluations.

Gartner recommends to understand the application development as "complex, adaptive system". A selection of metrics should consider this multidimensional behavior. Initial metrics proposed by Gartner for the development of IT Solutions are:

1. **Productivity**, Function points or KLOC per staff hour
2. **Quality**, Defects per function point or KLOC
3. **Cost**, Dollars per function point or KLOC
4. **Customer Satisfaction**, Scored survey based on client view of project results

In addition to these metrics, further metrics could be used. The number of the used metrics should not exceed ten in total.

An effort estimation at the beginning of a project was shown as a to be prone to errors. The reason is that only 60 percent to 70 percent of the real requirement will be known at this early project phase. The project should be re-estimated at the completion of each phase (in a waterfall model) or at completion of each iteration (if a spiral model is used). This way, the accuracy of the final estimate can be improved even more.

GartnerGroup statement for the introduction of metrics programs:

> *"Measurement programs that are instituted without a valid organizational context, and that are not used in specific SPI efforts, will fail at rates approaching 100 percent within three years (0.8 probability)"*

6 Summary

In accordance with the circle of the typical customers of the GartnerGroup, the metrics used focus more on the IT management viewpoint and less on a software developer. Especially process and resource metrics are considered by Gartner. Metrics which refer themselves to the established products of the software application preparation, such as models of design, are not

considered much. Furthermore, it can be found that the empirical models of evaluation of the employed metrics through Gartner are not disclosed.

References

[Gartner 1999a] Conference documents, Day 4, GartnerGroup Symposium/ITxpo99, November 1999, Cannes, France

[Gartner 1999b] The Monthly Research Review, Gartner Group, July 1999

[Gartner 1999c] Symposium/ITxpo Onside Guide, page 22-23,

Tool Description

ESTIMATION OF IT-PROJECTS
Highlights of a Workshop

Manfred Bundschuh
CFPS, President of DASMA, Bergisch Gladbach

Estimation is influenced by uncertainty and inaccuracy.

A prerequisite for estimation is an object to be estimated.

The greatest danger during Estimation is the management‘s persistant claim for a (too) early Estimation leading to the end that Estimation is mistaken for bargaining.

This as preliminary introduction to hint at the problemacy of estimation.

Before project start the estimation of effort, costs, dates and duration is **the base for profound planning** as well as measurement of project success. **estimations before project start** ask for according know-how-collections from corporate planning, where project risiks from market trends and according violation of trends as well as technological developments and Scenarios are availabale from Experiences of past projekt portfolios. There are no publicated investigations for this topic available up to date.

Estimation is a process influenced by conflicts due to resistance (not having a mind to estimateor to make a commitment, persons get evaluable), which leads to the question of **honesty of estimation.**

Estimations deliver figures for planning which can be tracked continuously. That is the base for measuring success. **It’s** p**aradox that project leaders don’t measure enough when evaluating their projects.**

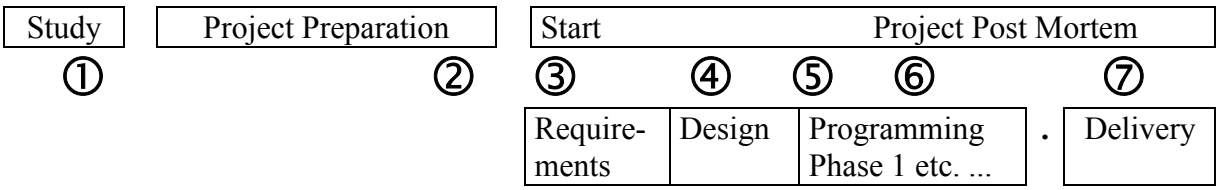
We learned from publications that (mostly larger) software projects exceed the effort or dates to an amount of 300 bis 1,500 %. In workshops and on conferences a deviation of 10 – 20 % - from the first estimation – is said to be a successful estimation.

1 Pragmatic Hints

Some **pragmatic hints** which help to sharpen the consciousness for the problemacy of estimation:

- The earlier an estimation – the larger ist the inaccuracy of the estimation.
- Every estimation is better than no estimation.
- The better estimations are documented – the better is the chance to gain experience in estimation.
- The more documented estimations are available – the better can be estimated.
- The more precise informations about an object to be estimated are available – the more precise can the estimation be.
- The estimated objects should be kept small and the work units independant.
- Mostly the communication factor will be forgotten.
- There don't exist 1:1 transferable formulae for estimation.
- Estimation should help in decision making and shouldn't end in itself.

The requirements for **controlling estimations** should be met, i.e., estimations should be repeated with growing knowledge during project progress (**tracking**), in order to actualize and precise the estimation and to document the experiences for future estimations. Only through such a consequent **management of estimation** can **expertise in estimation** be gained.




```
Legend:   ① = raw estimation
          ② = liable estimation(base for measurement of
              Project success)
          ③ ... ⑦ = tracking of estimation
          ⑦ = documentation of estimation experience
```

2 Basic Parameters of Estimation

From the above mentioned premisses can be learned, **that an estimation**

- **should be repeated**;
  A follow up estimation allows - with more precise informations – to estimate better. The comparison with the preceeding estimation delivers experiences for future estimations. A continuous tracking of the estimation allows to install an **early warning system for deviations** and supports transparency of actual changes (e.g. measurement of requirements creep, usually about 1 – 3 % each month of project duration).

- **should be performed in more than one variants**;
  The use of multiple estimation methods allows to compare estimations from diverseviewpoint, reduces the inaccuracy of the estimation and reinforces the security.

- **should be queried critically**;
  In any case the parameters of an estimation must be transparent because they strongly influence the result of estimation a priori. Developments in client-server environments or host programming with 4GL-languages must be estimated different from usual host COBOL developments; in large companies different from small firms with only few staff. It must be clearly documented e.g. for LoC methods (LoC = Lines of Code), if comments in programs are counted or - when using generators – the generated commands. Generally standards must be provided for time accounting, e.g. how many hours a person day, person month, person year has.

- **should be controllable**;
  Only controllable estimations give the chance to compare and allow a **feed oreward** (learning from past estimations for future estimations).

- **should be documented**;
  The main problem of estimation is not available documentation and hence the lack of Experience from past estimations. The better and the more estimations are documented – the more precise can estimations be and expertise in estimation be gained.

3 Rules of Thumb

Rules of thumb are <u>not exact</u> !

Rules of thumb shoudn‘t be used as liable standards !

Rules of thumb are easily applicable and can be used to ensure plausibility of estimations !

Rules of thumb are well known for their inaccuracy !

- A Function Point counter can count approximately 1,000 Function Points (**FP**) per day.
- The productivity of a programmer is approximately 5 FP pro person month (**PM**).
- 1 Function Point corresponds to ca. 100 LOC (**L**ines **O**f **C**ode; COBOL e.g.) - minimal 20 (OO, generators), more than 300 (Assembler).
- With 20 Project members a critical limit of team size is reached.
- Between 500 and 700 FP's the productivity ratio increases more than in other ranges.
- The quantity of Function Points increases about 1 – 3 % per month of project duration (creeping requirements scope), i.e. a project duration of years means that at the end of the

project 24 % more effort has been accomplished as there was demanded in the original requirements.

- The recomended quantity of test cases can be computed by the quantity of FP's exponentiated by 1.2.
- Each test case will be performed about 4 times during project progress.
- The error potential of a project consists of the sum of errors in all project phases, in coding, in user documentation and errors arising from corrections of errors.
- The error potential of a new development project can be calculated from the quantity of Function Points exponentiated by 1,25; enhancement projects: quantity of Function Points exponentiated by 1,27 (greater exponent due to latent errors in the base system).
- With each review, each inspection or each test step about 30 % of the existing errors can be found. Thus high quality can be reached by 6 to 12 of such subsequent quality measures.
- The staff required for maintenance of an application amounts to the quantity of Function Points divided by 500. I. e., one person can maintain and enhance (in small limits) about 500 Function Points (ca. 3 - 5 KLOC).
- The project duration (in calender months) is about the quantity of FP's exponentiated by 0,4.
- The staff required for a project amounts to the quantity of Function Points divided by 150 to 200.
- Project duration times quantity staff accounts to approximately the quantity of person months.

Source of information: **Capers Jones‘ books and his article „Software Estimating Rules of Thumb“, in the periodical IEEE Computer, "Software Challenges“ Column, July 28th, 1995**

3 Estimation Methods

Most estimation methods deliver as result a figure as measure for the size of the object to be estimated. Based on this a time relevant figure (effort) is elaborated from which costs can be derived. The total effort should be divided into the project phases according to a **percentage method**. The high esteem of the total effort from the actual effort of the first project phase due to the percentage method can parallel be used as comparative estimation.

There exist many known and valuable estimation methods. We learned from literature that the Function Point method is champion in comparison of the methods. Besides Cocomo (Constructive cost model, LoC-based) is in common usage. In principle this two approaches to estimate effort exist, based on requirements or program size.

There also exist tools supporting the estimation process as well as different estimation methods **(e.g. Checkpoint, Function Point Workbench etc.).**

The problems of the LoC Methods are thatLoCs are first available in a late phase of the project and that coding makes up only ca. 10% of the size of system development. When the coding phase is reached, there exist some KLoC (Kilo-LoC) methods for the estimation of the effort for component- and integration test.

The advantage of the Function Point method can be seen from the fact that meanwhile there exist some variants of the Function Point method: the **Data Point Method**, the **Object Point Method, Mark II,** the **Full Function Point Method** and **IFPUG 4.1**. IFPUG is the International Function Point User Group, which posted an einen internationalen standard for this most common used method which evaluates as object for estimation the requirements analysis document designed from users perspective).

The problem of the Function Point method is, that the requirements analysis documents are in early phases of the project not precise enough.

Important is the result of a study performed by Jeffrey, 1987, who found that the effort in projects up to a size of about 10 person years grows approximately linear and exceedingly exponential.

From the many estimation methods which can be found in literature the following are the better known:

- **The Pi times thumb method**
  The average from a worst case and a best case estimation is computed. A variant is – A quarter of: the worst case plus best case plus 2 times the average.

- **The Analogy method**
  Comparison of size in LoC of project post mortems.

- **The Relational Method**
  Comparison of indices of project post mortems, e.g. COBOL=100, Assembler=130, L/I=85 or Skill = 100, 120 oder 90.

- **The Weightiness method**
  Estimation with formulae which give different weights to different parameter and / or phases of system development.

- **The method of parametrical estimation equations**
  Estimation with formulae for parameters which strongly influence the effort of system development, e.g. the formula of Putnam (SLIM = Software Life Cycle Management).
- **The Multiplicator Method**
  The average productivity of programmers in LoC is multiplied by the estimated LoC, e.g. the Wolverton method.

- **The Percentage Method**
  The effort is relatively divided into the phases of system development, e.g.:

Phase No.	Percentage	Phase	or	Phase No.	Percentage	Phase
1.	10 %	Requirements Analysis		1.	11 %	Requirements Analysis
2.	30 %	Requirements Specification		2.	11 %	Anforderungs-Specification
3.	30 %	DP-Concept		3.	5%	Logical System Specification
4.	25 %	Coding		4.	10 %	Physical Design
5.	5 %	Delivery		5.	46 %	Coding and Module Test
				6.	5 %	Implementation
				7.	12 %	System Test

- **The IFA-PASS-Method**
  A method based on the waterfall modell and according results of the firm IFA-PASS (Züricher Institut für Automation).

- **The Wolverton Method**
  A derivate of the Multiplicator Method which is also dependend upon the difficulty and the type of software.

- **The Cocomo Method**
  A three step LoC based method, developed by Barry Boehm from TRW.

5 Variants of the Function Point Methode

All Function Point Variants consist of 3(4) Parts:

- Estimation of entities (objects, functions, transactions)
- Weighting of adjustment factors
- Computation of effort according an „Experience curve“
- (- Computation of an experience curve before introduction of the method)

Following is a short characteristic of each Function Point variant.

- **The IFPUG 4.0 Method**
  Informations about the IFPUG 4.0 Function Points is available in the following Internet URL: http://www.ifpug.org

- **The Data Point Method**
  Developed by Harry Sneed from SES (Software Engineering Systems) for estimation based on data objects.

- **Die Object-Point-Method**
  Developed by Software AG for estimation of Object Oriented System Development.

- **Die Mark II Method**
  Method developed by Mark Symons in England 1988 for the estimation of effort in 4GL environments.Mark II uses less parameters than IFPUG 4.0 and is easier to use. In smaller projects there are slight differences to IFPUG 4.0 when measuring the project size Mark II was developed according Mark Symons,
  - to reduce the subjectivity by counting the quantities of the entities instead of the data sets;
  - to measure the development effort instead of the delivered functionality;
  - to add 6 complexity factors to the 14 VAF‘s.

- **SPR Function Points**
  SPR Function Points are a simplification of IFPUG 4.0. The complexity of data objects and transactions ist counted as average thruoghout and the VAF is based on 2 instead of 14 factors.

- **Problem Complexity and Data Complexity**.
  A specialty of the SPR Function Points is the **approximation** of Function Points if only Parts of the entities are known, they thus can be used,
  if only thw ILF's (Internaal Logical Files) or
  only the EIF's (External Interface Files) or
  only the EI's (External Inputs) or
  only the EO's (External Outputs) or
  only the EQ's (External Inquiries) are known.

- **Feature Points**
  Feature Points are also a simplification of IFPUG 4.0. They were developed 1986 by Capers Jones regarding the complexity of the system development process. Besides the data objects (which contribute less) and transactions 6 entities measure the **algorithms**.The VAF is also based on 2 instead of 14 factors. The problem of the method is, that The definition od algorithm is not precise enough to use it as standard.

- **3D Function Points**
  3D Function Points were developed 1989 by Boeing Computer Services. They measure system development by data, functions and control flux. Data are measured according IFPUG 4.0. The quantity and complexity of functions is added as well as the quantity of control statements (system states and state transitions).

- **Full Function Points**
  Full Function Points were developed during the last years by the Software Engineering Management Research Laboratory (UQAM) of the **University in Quebec,** for technical and scientific Applications. It consists of two parts: Function Point counting similiar to IFPUG 4.0 and a measurement of the complexity of functions. This is a compromise with project leaders‘ complaints about this topic. A public domain version of the FFP measurement manual is available in the following Internet URL:
  http://www.lrgl.uqam.ca/ffp.html oder http://www.lmagl.qc.ca.

- **Function Point Prognosis**
  During the FESMA 1998 Conference in Antwerp Manfred Bundschuh from CNV AG (IT of AXA Colonia Insurance) presented his research of regression analysis on

project estimation data based on about 20 projects. During the FESMA 1999 Conference in Amsterdam he reported that the former results proved to be consistent and only slightly different, based this time on about 40 counts. His main result was, that there is a strong correlation between the sum of the quantity of EI's and EO's (he called IO) and the unadjusted Function Points. He examined the error and found about 15 % deviation. Since estimation is influenced by inaccuracy per se, his formulae for Function Point Prognosis can be used early in system development when only the quantities and not the function points are known, in order to estimate the Function Points. Here are the formulae:

CNV Prognosis		Formula	R^2
1998		FP = 7,3 IO + 56	0,9525
1999	**Total**	**FP = 7,6 IO + 39**	**0,9509**
	PC	FP = 6,5 IO + 134	0,9760
	Host	FP = 7,8 IO + 11	0,9580

The 1998 paper is public available in the folowing Internet URL: http://www.gm.fh-koeln.de/~bundschu, there in Vorträge, FESMA-Vortrag, Antwerpen 1998.

Also the following function proportions (collected by Manfred Bundschuh from different literaturein his FESMA 1999 report) can be used for a Function Point prognosis:

			EI	EO	EQ	ILF	EIF
Average Function Complexity	**IFPUG**		**4**	**5**	**4**	**10**	**7**
	ISBSG Rel. 5	**Total**	**4,3**	**5,4**	**3,8**	**7,4**	**5,5**
		Asia Pacific	4,0	5,6	3,9	7,4	5,6
		Europe	4,2	4,9	3,8	7,2	5,3
		North America	4,9	5,2	3,8	7,6	5,5
	CNV 1998	**All**	**4,6**	**5,5**	**4,3**	**8,0**	**5,9**
		PC	4,1	5,7	3,9	7,1	5,3
		Host	4,9	5,5	4,6	8,5	6,1
	CNV 1999	**All**	**4,6**	**5,7**	**4,3**	**8,2**	**6,1**
		PC	4,0	5,7	3,9	7,3	5,4
		Host	4,8	5,7	4,5	8,5	6,2

The following Function ratios (also collected by Manfred Bundschuh from different literature in his FESMA 1999 report) deliver another possibility for Function Point Prognosis:

Function Ratios (Percentage)	Number of Projects	EI	EO	EQ	ILF	EIF
CNV 1996	8	34	35	11	18	2
CNV 1997	12	18	43	12	18	9
CNV 1996/97 = FESMA 1998	**20**	**27**	**39**	**11**	**18**	**5**
CNV 1998	19	21	41	13	15	9

Tool Description

CNV Σ (1996-1998	**39**	**25**	**39**	**14**	**17**	**6**
ISBSG Rel. 5	451	37,2	23,5	13,2	22,3	3,8
Metricviews		26-39	22-24	12-14	24	4-12
Checkpoint		20	24	10	43	3
Nigel Scrivens		33	21	19	23	4
ISBSG Rel. 5 Enhancement Projects	119	36	32	12	15	5
Average of this table:		**28,6**	**26**	**13,8**	**20,1**	**5,8**

- Unser Arbeitskreis Softwaremetriken der GI-FG 2.1.9 ist ab sofort eine

  Fachgruppe Software-Messung und -Bewertung (FG 2.1.10).

  Der vorläufige Sprecher dieser FG ist der bisherige Leiter des Arbeitskreises, *Prof. Dr. Reiner Dumke*, Universität Magdeburg.

DASMA Spring Conference 2000

15 / 16 March 2000

probably in Düsseldorf, Germany

DASMA is the German speaking user association for software metrics and effort estimation, with the aim of supporting the co-operation of software developers and users from science and practice, contributing to the development of standards for software metrics, and cultivating the relationships to other international metric organisations.

The scope of the conference will be the exchange of experience about

* software sizing
* effort estimation
* risk management
* benchmarking

between experienced experts from industry and science as well as interested newcomers. The focus of the conference is on the practical application of these subjects in software projects.

Conference Programme:

Wednesday, 15 March 2000

* **IFPUG 4.1 CFPS Certification Examination**
* Workshops organised by FESMA

Thursday, 16 March 2000

* Specialist presentations
* Experience exchange

The official conference language will be German, the IFPUG Certification and the FESMA Workshops will be held in English. Participation in the IFPUG 4.1 CFPS Certification Examination is also possible without attending the conference. For further information, please visit the DASMA web page

http://www.dasma.de

or contact the DASMA board member Iris Koll, telephone +49 - 911 / 519550

The Performance Engineering Maturity Model

Andreas Schmietendorf, André Scholz
University of Magdeburg, Faculty of Computer Science

Tool Description

ABSTRACT

This contribution presents a model for evaluating the level of integration and application of performance engineering, which is called performance engineering maturity model. It leans against the well established capability maturity model from the software engineering institute SEI.

1 Introduction

In practice software engineering most time only considers functional specifications. But meanwhile companies pays more and more attention to non-functional requirements. Therefore processes are adapted in order to develop high quality software and quality assurance systems have been installed within existing processes.

One of the most critical non-functional quality factors is the performance characteristic of a software system. Performance can be defined as the capability of a system to process a given amount of tasks in a determined time interval. Thereby, performance of software systems stands in direct relationship with the speed of accompanying business processes. This relationship appears especially important in the context of the future market 'Electronic Commerce with the help of internet technologies'. The Gartner Group evaluated this market this year as the most important expansion market [1]. There has to be a special attention to the right integration of customer's, subcontractor's and service provider's business processes with regard to performance aspects.

A lot of developing projects as well as productive systems fail because of insufficient performance characteristics. Up to now performance characteristics are often only considered at the end of the software development process. If performance bottlenecks are only recognized at the end of the software development, extensive tuning activities, design modifications of the application or complete rejectings of entire software systems will be necessary.

That’s why a performance oriented software development method, like performance engineering (PE), should be integrated within the engineering process [2]. PE considers the response time as a design target throughout the whole software system development process. Response time targets and analyses of the respective design structures are continuously compared. In early stages response time metrics can be quantified using estimation formulas, analytical models, simulation models and prototypes. Deviations lead to an immediate change in the software design. Thus, PE guarantees an adequate response time behavior of the productive system. PE of software systems needs an entire approach considering the complete software development process. For this purpose, practical models of integration are already available [3].
The efficiency of the PE application depends decisively on the maturity of the PE integration and application. The higher the integration of PE processes and methods, the lower the PE costs and the performance entailed development risks. For that reason, we propose a performance engineering maturity model (PEMM), which lean against the capability maturity model (CMM) to determine the grade of maturity of the PE process within the software development.

After an introduction of the basic framework of the CMM, the PEMM and it's practical application is explained.

2 The Capability Maturity Model

The quality of software systems depends decisively on the quality of the corresponding software engineering process. That's why a software buyer is interested in getting to know the grade of maturity of the vendors software engineering process to draw conclusions on the software system's quality.

The Software Engineering Institute (SEI) of Carnegie Mellon University developed a framework for evaluating the maturity of a company's software engineering process in 1987 by order of the US Department of Defense. This framework, which is known as the Capability Maturity Model (CMM), distinguishes five different maturity levels. With the help of an evaluation catalog, the maturity of a company's software engineering process can be assigned to one of these levels. The levels are based on each other, which means that if an engineering process fulfills the requirements of a level, it also fulfills all requirements of all levels below. With an increase of the CMM level the development risk can be reduced and the productivity of development as well as the quality of the product can be increased. The individual levels can be described as follow [4]:

- *Level 1:* The structure of the software engineering process is only established informally. The process is very chaotic and unstructured. The arising costs and quality of the system can not be predicted. Schedules are difficult to determine. About 74% of all US-companies are at CMM level one.
- *Level 2:* The process can be repeated intuitively, because the requirements and the structure of the process already exist basically. There is a reasonable control of schedules. The decisions on when to use which methods are still done informal and in an ad-hoc manner. That's why the predicted costs and quality of the software are still very variable. About 22% of all US-companies are at CMM level two.
- *Level 3:* The whole software engineering process is well defined. The prediction of schedules and system costs are precise. The quality of the system has improved, but can not be determined accurately. Individual process steps are assigned to organizational responsibilities. A study has shown that only 4% of all US-companies have a software engineering process, which can be assigned to level three.
- *Level 4:* The engineering process is managed. Metric systems are used to ensure the software quality. Thus, quality factors are determined by metrics, which have to be quantified. An organizational unit uses these metrics to control the process as well as the software system. The unit is in the position to step into and adapt them. Furthermore, metrics are also applied on productive systems to reflux the data to further engineering projects.
- *Level 5:* The process is optimized and has reached the highest level of maturity. There are methods and concepts for a continuos process improvement and automation. The process has the ability to adapt itself to other development techniques and technologies.

Tool Description

The SEI proposes to determine the maturity level with the help of assessments. A criteria catalog (table 1) describes the key criteria, which are necessary to qualify for a maturity level. The fulfillment of each key criteria is examined by questionnaires. The whole assessment is based on an 85-item questionnaire. The questions have to be answered with 'yes' or 'no'. Some of these questions are key questions, which means that their fulfillment is essential for reaching a specific CMM level. 80% of all questions and 90% of all key questions assigned to a specific level need to be answered with 'yes' to qualify for a maturity level.

Level 1	Initial	*no criteria necessary*
Level 2	Repeatable	Requirement Management
		Project Planning
		Project Tracking and Controlling
		Subtask Management
		Quality Assurance
		Configuration Management
Level 3	Defined	Process Organization
		Process Definition
		Training Schedules
		Integration of Software Engineering Management
		Software Product Engineering
		Coordination of all involved Groups
		Early Error Removal
Level 4	Managed	Quantitative Process Management
		Quantitative Quality Management
Level 5	Optimized	Error Avoiding
		Innovation Management
		Process Improvement Management

Table 1: Maturity Level Key Criteria

The presentation of the whole questionnaires would break up this contribution. We refer to the relevant literature [5].

A few aims are pursed by evaluating the maturity of the software engineering process. The quality of processes should be made transparent. The market should have an objective comparison criteria. Higher maturity levels induce a more efficient application of software engineering. Predictions and the accuracy of models are getting more precise. Thus, costs and time spent for software engineering can be decreased. So, existing requirements with regard to costs, schedules and product quality are easier to fulfill.

The CMM focuses on the software engineering process only. It was fixed a few years ago. Additional accompaniment concepts of software engineering, like performance engineering, are not considered within this model. Thus, there result two possibilities for considering performance engineering processes. On the one side the CMM could be extended or on the other side a new model could be created. We propose to create a new maturity model for performance engineering, because some companies still don't use performance engineering concepts, because they even develop performance uncritical systems.

3 The Performance Engineering Maturity Model

It's the aim of the PEMM to evaluate PE processes as well as the process integration [6]. The proposed evaluation model can be used to evaluate an organization and to use it for their own specific further process development. Further on, the PEMM level can become a selection criteria for choosing a software system provider for critical or semi-critical products. Thereby a PEMM level states to what extent a concrete organization is in the position to carry out a performance oriented software development. Thus, a system provider is in the position to stand out against the market.

The Goal-Question-Metric method (GQM) was used for identifying wise questionnaires and metrics. In a first step suitable aims have to be defined. Then suitable questions have to be selected and then necessary ordinal measures (at present only 'Yes' or 'No') need to be determined for a quantified answer of these questions [7].

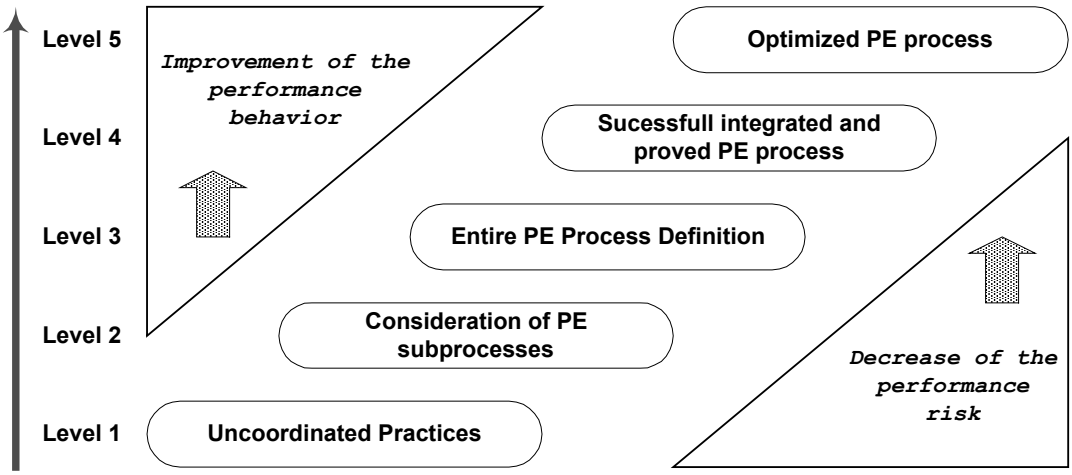


Figure 1: Maturity Levels of the Performance Engineering Processes

In the following every individual level of the PEMM (see figure 1) is described by its most important characteristics in three sections. First, a description determines the general contents of the level. Similar to the CMM, PEMM key criteria were defined for every maturity level, which are the elementary basis of the level. Several aspects are assigned to every key criteria, which show which tasks have to be done for the fulfillment of the key criteria. Questionnaires can be derived from these aspects. In this contribution selected questions from the fields of organization, project management, process management and technology are listed. However, by considering the whole questionnaires the entire model is much more complex. Because of the respective scope the presentation of all questions isn't possible in this contribution.

All levels are based on each other. A respective level implies the description and process maturity of all subordinate levels.

3.1 PEMM Level 1 - Uncoordinated Practices

Tool Description

(a) DESCRIPTION:

- The use of PE depends on the personal engagement of individual developers.
- The organizational structure does not support the PE process explicitly.
- Accordingly, individual methods are only used unstructured.

(b) KEY CRITERIA: DO NOT EXIST

(c) EXAMPLES FOR A CATALOGUE OF QUESTIONS:

PEMM Level 1 is the initial stage for all companies. Therefore it's not useful to define questions to determine this level.

3.2 PEMM Level 2 - Consideration of PE Subprocesses

(a) DESCRIPTION:

- Parts of the whole PE process are already considered.
- Individual PE service provider exist.
- However, a complete process description is not yet available.

(b) KEY CRITERIA:

- **Performance Requirement Management:** Performance characteristics of essential system functions have to be defined, which are required from the customer.
- **Performance Tracking:** The performance characteristics need to be verified throughout the whole life cycle of the information system.
- **Personal Identification:** All engineers, which are involved in development and in maintaining, are obliged to the quality factor performance.

(c) EXAMPLES FOR A CATALOG OF QUESTIONS:

Perspective	Questions	yes	no
Organization	• Is there a fundamental management-agreement that performance should be considered within the development process? • Is there a performance-related communication channel?		
Project Management	• Are there enough resources (personnel, infrastructure) to do PE? • Do the staff have the necessary skill to do PE? • Does the project manager know elementary concepts of PE? • Are performance-related tasks delegated within the project?		
Process Management	• Are single PE procedures completely defined in writing? • Is a PE plan for each project created?		
Technology	• Are single tools used for PE tasks?		

	• Are there already performance experiences with technolo-gies in use?		

Table 2: Level 2 - Examples of Questions

3.3 PEMM Level 3 - Entire PE Process Definition

(a) DESCRIPTION:

- The PE process is considered within the entire software development process. All available PE methods and tools are used comprehensively with regard to the existing performance risk.
- Performance-relevant product and resource metrics are selected for the PE use and standardized within the organization. These metrics are stored managed in appropriate database systems to guarantee a continuous reflux of experiences.
- The performance requirements of customer, which are defined in the system analyze phase, are used as success criteria at the final inspection test. Furthermore, they are arranged in service level agreements (SLA) with the provider of the information system.
- Furthermore, an initial organizational structure for the entire PE process has to be defined and introduced step by step in level 3.

(b) KEY CRITERIA:

- **Definition of PE Processes:** There is an entire definition of all processes which are necessary for PE. Different levels of abstraction have be considered.
- **Performance Problem Prevention:** Performance related problems as well as PE related costs are recognized very early.
- **Performance Aim Management:** Engineering tasks are focussed on performance aims, having an equal position like functional requirements.
- **Performance Engineering Management:** The tasks of PE are assigned to organizational structures. The whole process is coordinated by a management.
- **Metric Initiation:** while there are already performance metrics, metrics of other paradigms have to be introduced to ensure a high level a quantifying and the ability to evaluate existing processes and used resources.

(c) EXAMPLES FOR A CATALOG OF QUESTIONS:

Perspective	Questions	yes	no
Organization	• Is there an instance, which is responsible for improving and adapting the PE process? • Is the training of project members in PE methods fixed by an organizational instance? • Is there an independent instance, which controls the correspon-dence of performance analyses and determined standards?		
Project Management	• Do wrong performance characteristics lead to an immediate consideration within the software engineering?		

	• Are there coordination mechanism, adjusting and scheduling single activities?		
Process Management	• Is the whole PE process defined and documented? • Is the integrated PE process the standard process within the software development?		
Technology	• Is there an extensive tool support for all PE methods within the whole life-cycle? • Is there a policy to only use standard PE tools?		

Table 3: Level 3 - Examples of Questions

3.4 PEMM Level 4 - Successful integrated and proved PE Process

(a) DESCRIPTION:

- The PE tasks are a firm part of the software development. Thus, they are integrated in respective process models. Process, product and resource metrics, which are introduced in level 3, lead to extensive empirical experiences.
- All employees from the developer and service provider, which are involved in PE processes, have access to performance relevant metrics and experience data, considering different security and view properties.
- Metrics are used for estimations of characteristics (Rules of thumb), for performance models or for statistical evaluations. Furthermore, it should be possible to estimate the costs of PE.
- By the gradual increase of experience and a decrease of performance problems while implementing information systems, the surplus value of PE can be understood directly.
- Furthermore, domain specific instances of PE are defined, e.g. for software systems, tools and technical applications. In this way, complex systems and new tools can still be controlled in future.
- The organizational structure was developed further in accordance with the experiences.

(b) KEY CRITERIA:

- **Coordination of the Reflux Circle:** Information flows are established between the developer and the service provider to exchange performance-relevant experiences and metrics. This reflux circle is supported on this level by repositories or metric databases.
- **Metrics based PE Controlling:** The surplus value of PE can be proved on the basis of saved costs.
- **Performance Transparency:** All used components in different layers of the software system have a sufficient description of their performance characteristics and resource consumption.
- **Quality Control:** The development product as well as the necessary components are subject for a continuos quality control. Thus, customer requirements with regard to performance can be reached

(c) EXAMPLES FOR A CATALOG OF QUESTIONS:

Perspective	Questions	yes	no

Organization	• Are organizational structures assigned to the comprehensive PE process?		
Project Management	• Is there a cost-benefit analysis to determine the surplus value of PE?		
Process Management	• Can the PE process immediately be adapted to new technologies and new fields of application? • Are metrics used to compare and evaluate performance characteristics?		
Technology	• Does the technology infrastructure support the rapid customization of the process? • Are performance metrics stored in a database, which is available within the whole development and maintenance process?		

Table 4: Level 4 - Examples of Questions

3.5 PEMM Level 5 - Optimized PE Process

(a) DESCRIPTION:

- The maximum degree of process maturity is achieved.
- PE can be applied to all fields of operation.
- Technological modifications within the software development like the use of a new middleware can be absorbed by PE, too.

(b) KEY CRITERIA:

- **Innovation Management:** The PE process is adaptable. Experiences from new application domains as well as new research results with regard to methods an tools flow continuously in the optimization of the process. The process can be adapted to new requirements, so that customer demands in still unknown domains and technologies can be realized with ensuring a determined performance characteristic.

(c) EXAMPLES FOR A CATALOG OF QUESTIONS:

Perspective	**Questions**	**yes**	**no**
Organization	• Is there a continuous PE improving process system?		
Project Management	• Are new and promising methods automatically tested and integrated?		
Process Management	• Do improved engineering techniques lead to a revision of the software engineering process?		
Technology	• Are tools able to be used in all kinds of PE applications?		

Table 5: Level 5 - Examples of Questions

4 Practical Aspects of Application

4.1 The PEMM - A strategic Target Figure

For a practical application of the PEMM it's useful to restrict the initial application of the evaluation model to a manageable time frame. The model, which is shown in figure two, only consistent of the core of the evaluation model with four levels. Level four should be achieved in five years. In our experiences , this is a typical time frame for strategic plans, which also can be appreciated by the management. Respectively the temporal sequence, which describes when is which step reached, can be understood as a master plan with corresponding mile stones.

Not only statements of the temporal horizon of the achievement of a maturity level, but also cost based statements are necessary for a technical application of the evaluation model:

ROI „Return on Investment“: This critical measures is the basis of decision for the management of a respective company. It indicates them, if PE concepts should be initiated or improved.

Infrastructure based Costs: These costs comprise the costs for the creation and maintaining of the measuring and modeling instruments, e.g., the creating of a benchmarking laboratory or the implementation of databases for storing performance metrics.
Personnel Costs: The complex tasks of PE require high specialized employees. Costs are induced by a continuos training as well as by the project tasks itself.

Performance based Development Risk: The performance based development risk should reduce in the same degree as the PE maturity level is increasing.

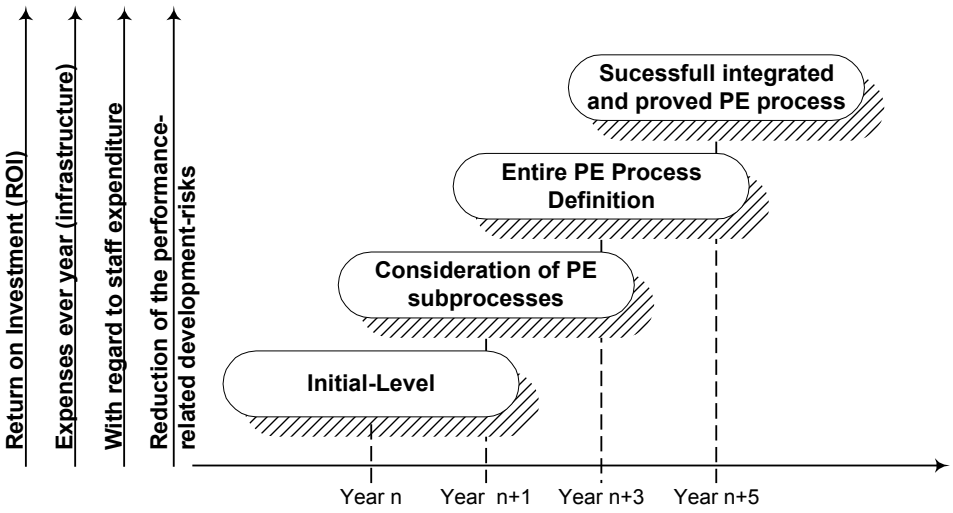


Figure 2: Adapted Performance Engineering Maturity Model

4.2 The Evaluation Process

Obviously, the expenditures for the determination of the maturity level, because of the answering and analyzing the questionnaires, should be minimized. This can be ensured by a tool based process evaluation or by an integration of the evaluation process within the evaluation process of the CMM, if the organization is intending to determine its CMM level, too. By that questionnaires should be designed in a way, that organizational units are only polled once.

The polling itself should be processed in a predefined and standardized procedure. The real evaluation of the present state, which is the basis of a strength and weakness analyze of the respective organization, is taken place with the help of this procedure. A catalog of measures, which consist of concrete tasks for the achievement of the next PEMM level, have to be defined on the basis of these results. The polling can be parted into the following phases:

- **Preparation:**
  - Order by the management
  - Collection of information
  - Training
  - Enabling trust
- **Realization:**
  - Polling different groups
  - Evaluation
  - Strengths weaknesses profile
  - Explication of potentials for improvement
  - Discussing the results
- **Reworking:**
  - Description of the present and rated situation
  - Description of the strengths weaknesses profile on different topics, e.g., phase or technology based

5 Conclusions and Outlook

We propose a performance engineering maturity model for evaluating the application and integration of PE processes. Thus, the IT management has an instrument for a continuos comparison and improving of these specific processes. The model supplies systematics and guidelines for a process improvement and identifies existing weaknesses. These advantages become more important when it becomes clear that processes have a higher potential for improvement with regard to the whole development task than methods and tools.

For that reason we have the vision, that the PEMM can reach a comparable field of application as the CMM already has. IT software system projects with strict performance

requirements, especially electronic commerce applications, software systems within the telecommunication or within military applications, are now able to consider the PEMM level within a contract. The inclusion of real time systems is not recommended at this time, because the development of such kinds of systems are based on very specific engineering processes and methods. With it some developing companies will be not longer in the position to take part on invitations to bid, if they don't improve their performance engineering processes. This assures the customer, that companies, which come in range, develop a solution with the necessary performance requirements at a given duration of time and a fixed price.

But unfortunately some aspects still remain problematic while using the PEMM. The structure and the extension of the questionnaires are very sensitive. Empirical evaluations, if the PEMM levels are sufficiently refined by the questionnaires despite the binary set of answers, are still missing up to now.

Furthermore, the theoretical model has to be moved in his practical application. In near future an evaluation form with all necessary questionnaires will be provided on the internet. Thus, every IT manager is in the position to rank his department with regard to the PEMM level. The results will be analyzed in a further study.

References

[1] GartnerGroup: Tagungsband, Symposium/ITexpo99, Cannes, Frankreich, Nov. 1999

[2] Scholz, A.; Schmietendorf, A.: Zu den Aufgaben und Inhalten des Performance Engineerings, in: Informatik Spektrum, erscheint Anfang 2000.

[3] Scholz, A.; Rautenstrauch, C.: Vom Performance Tuning zum Software Performance Engineering am Beispiel datenbankbasierter Anwendungssysteme, in: Informatik Spektrum, Bd. 22, Hf. 4, 1999.

[4] Balzert, H.: Lehrbuch der Software-Technik. Software-Management. Software-Qualitäts-sicherung. Unter-nehmensmodellierung, Heidel-berg, Berlin, 1998.

[5] Chrissis, M.; Curtis, B.; Paulk, M.: Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Technical Report-24, Feb. 1993.

[6] Scholz, A.; Schmietendorf, A.: A risk-driven Performance Engineering Process Approach and its Evaluation with a Performance Engineering Maturity Model, in: Proceedings of the 15th Annual UK Performance Engineering Workshop, Bristol, UK, July 22-23, 1999.

[7] Dumke, R.; Foltin, E.; Koeppe R.; Winkler, A.: Softwarequalität durch Meßtools, Vieweg-Verlag, Wiesbaden 1996

Beginning a Measurement Program - A Learning Experience with Function Point Analysis

HERBERT BIERFERT

BONNDATA GMBH, BONN

ABSTRACT

Bonndata delivers software and IT services to a large German insurance company and has a staff of about 500 people, software engineers being the largest group. In 1997 we started a metrics program in our company which aimed at establishing a business view upon our software development activities and at controling our improvement efforts from a financial point of view. We first wanted to benchmark us against other companies in the field of financial services. Our hope was that Function Point Analysis (FPA) would give a common base for performance measurement, cost estimating, and software portfolio analysis. When we tried to implement FPA in a variety of different subcultures we got into several dilemmas from which we could not escape and recover. We had to abandon the endeavour and restart again under totally different premises. The main point of this paper is that FPA may not be a tool suitable for beginning a metrics program, because it focuses upon a special technology of functional size measurement instead of establishing a context for gradually incorporating measurement into the management of software processes. We learned that no single software measure - function points or other - was really important in the beginnings, but that better managing the processes of estimating, planning, and controling was what we needed in the first place. These processes determine which measures are useful or useless. When restarting the measurement project in 1999 we first explored the field of measurement and process improvement in many directions. Right now we are entering the mainstream of process improvement and following the path of the Capability Maturity Model (CMM).

1 Introduction

Function Point Analysis (FPA) [1] responds to very concrete needs. In 1997 we chose FPA as the starting point for our measurement program [2], because it focusses on the business value of software and upon the users‘ interest in the functionality of the software. Clearly, compared to the software technology-centric search for the ultimate language, tool or development methodology its promise is a wider focus upon business needs and what really counts, the results for the user. At the same time the amount of data for measuring performance and progress should be limited for reasons of resources necessary to collect and interprete them. FPA fits neatly in this picture, too. Last but not least FPA appeared to be something like an industry standard and was supported by an international user group (IFPUG) and a national metrics groups (DASMA). We had the support of some of the best specialists available in Germany [3].

We started with measuring 6 projects which had been completed soon before and collected the data necessary for external benchmarking. The importance of FPA for our program demanded that the method, its prerequisites and its application could be explained unambiguously to our very sceptical software engineers. But the more we strived to justify the implementation and application of FPA for benchmarking purposes, the more we got unsure ourselves that we were following the right path to controling our improvement efforts. After the application of FPA to three other ongoing projects we had to admit that our results did not withstand a

critical evaluation and had not come close to what we had expected. Here are some of the reasons that seem to have caused our failure:

- Inadaquate foundations of FPA, especially unclear relationships to modern modeling concepts, to modern architectures, and to software sizing (e.g. LOC's as opposed to functional sizing)
- The complexities of our different projects, environments and cultures, ranging from maintaining legacy systems to high-risk projects following the modern trends in architecture, language, and life cycle
- Missing coordination with the established means of reporting and controling (which - needless to say - hadn't been working to our satisfaction)

These are very different categories. We will take them up in turn. The discussion is not meant to advance the state of Function Point Analysis or to reveal any unknown facts about FPA. We are interested in the response of our collegues to FPA and what this response tells us about the prerequisites and the applicability of FPA in our company and similar ones.

2 Functional Size and Application Types

The measurement of functional size is a hot topic. Several schools of thought seem to fight and to unite at the same time. Such things belong to any effort on an emerging technology and we are not going to complain about the situation but hope that eventually some common progress will be made [4]. Our issue here is suitability of FPA as a cornerstone for introducing a metrics program. The question is whether covering a diversity of modeling practices, architectures and project types with one counting technique is a practical thing to do and can deliver early and credible results for a beginning measurement program.

We were confronted with different kinds of functional modeling and different degrees of maturity in modeling practices. While one project had been using object-oriented modeling techniques, others had been using traditional ER modeling. The rest did not use any models beyond physical data models [5]. Under the common name of FPA we were soon doing quite different things: translating, creating or reconstructing functional models. When we found ourselves teaching functional modeling the question arose whether we should first teach an established technique like ER modeling oder whether we should take the shortest path to counting FPs according to IFPUG rules. On the other hand people in those projects already using established techniques wondered why they should transform their models to an apparently similar approach just for counting according to rules which appeared to be so crude that the differences in modeling concepts didn't seem to matter much.

In order to proof the usefulness of FPA right from the beginning we had included the projects that mattered most to our company, some in the range of 1000 FPs and beyond. One project was to reengineer a mathematical subsystem, another had developed a point-of-sale system, yet another had programmed a system to manage financial assets. The types of projects were quite different and reflected the services our company had to deliver to a large German insurance company. The architectural approaches differed likewise. A lot of questions regarding the appropiateness of the FPA method under these different circumstances could not be answered satisfactorily. Sometimes we had to tailor the method in order to handle only a portion of a system that should be counted and we got into trouble when trying to decide on

the boundary of the system and applying the notion of the USER. Swearing that the function points reflected the view of the business user didn't help much to give the numbers the necessary empirical evidence.

None of the groups felt that their effort was worthwhile and that the conclusions regarding their productivity as compared to each other were justified. The individual conditions were too diverse. The metrics team found itself investing far too much time in preparing and training the projects for counting function points. This partly reflected the weakness of the method - which may be overcome in the future - and partly our own situation of different subcultures. This is an inherent phenomenon and an obstacle to introducing uniform quantitative methods across an organization starting a metrics program.

3 Estimating

To make a discussion of the next group of findings more meaningful and to make our interpretation of these findings plausible, a high-level causal model of software development will be used.

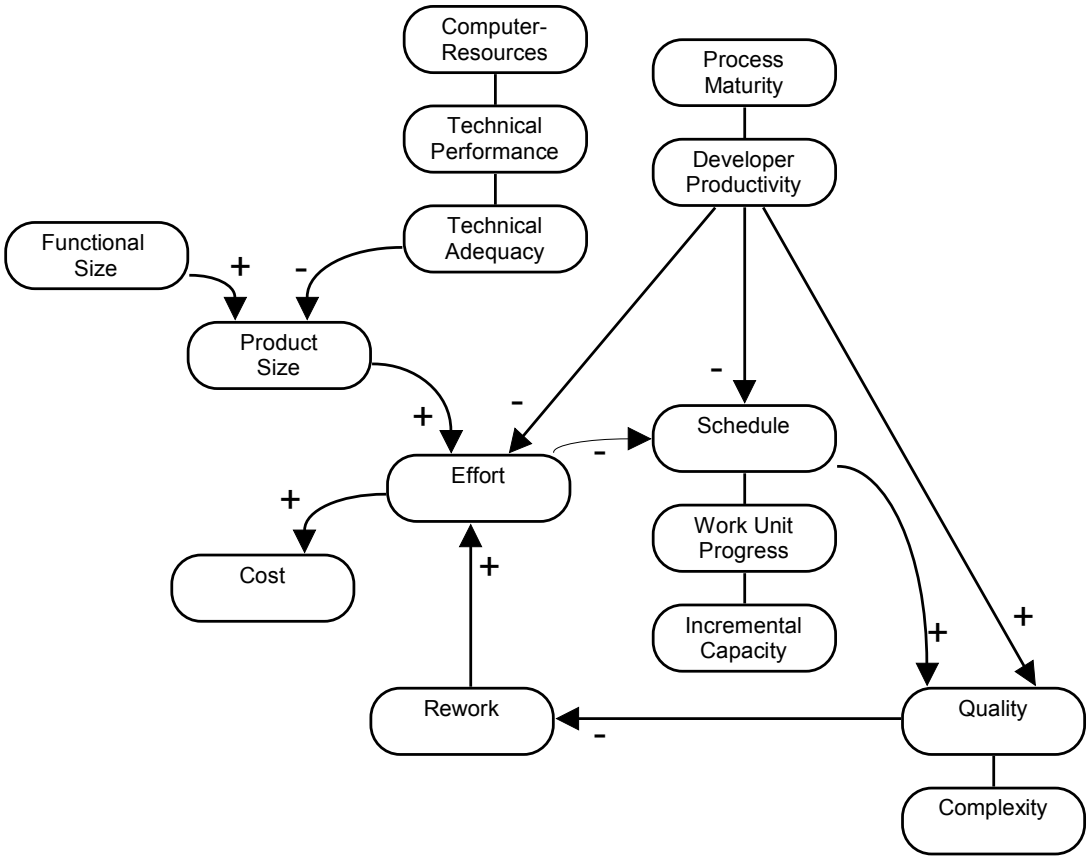


FIGURE 1: **A CAUSAL MODEL OF SOFTWARE DEVELOPMENT (ADAPTED FROM "PRACTICAL SOFTWARE MEASUREMENT. A FOUNDATION FOR OBJECTIVE PROJECT MANAGEMENT." OFFICE OF**

THE UNDER SECRETARY OF DEFENSE FOR ACQUISITION AND TECHNOLOGY. PART 4. [6])

The diagram includes important elements of the software process. It can be easily interpreted, with arrows between the elements read as ”influences”. The nature of the influence is often complicated, but the plus and minus symbols along the arrows give some general guideline. If quality increases, rework decreases. If schedule increases, quality increases. Only the relationship between immediate neighbors should be considered when interpreting the symbols. For instance, if effort increases, the schedule required to do the work decreases, though not under any circumstances and provided that the amount of functionality and rework remain constant. Some relationships between closely related elements are not specified.

Of course, the diagram highlights only those relationships which are considered to be worthwhile for the given purpose, i.e. measuring the software process. Each node represents some important issue and may contain a variety of actual measures. Note the causal loop in the lower part which refers to the well-known phenomenon of feedback loops in processes. The diagram illuminates the dynamic nature of software development. The delays between the presence of a problem and the visibility of its effects elsewhere may be in the range of many months. Cost is controled by literally any parameter in the model. FPs are one special measure for functional size.

When the use of function points is advocated this is very often accompanied by a critique of using LOC as a measure for the size of software, as an estimator for effort and as a denominator for performance. We, too, were impressed by this critique initially but grew suspicious that there is more to it than using one or the other. The above dynamic model is supposed to explain from a different angle why our expectations regarding the introduction of FPA were far too high. At the same time it explains the fact that under very special circumstances function points may indeed fulfil its promises and serve for the above purpose of estimation.

First of all, the model highlights the non-controversial fact that functional size and product size are related, product size being the amount of code that actually has to be worked out by hand. Product size is, however, heavily influenced by technological factors like use of generators, reuse and frameworks, among others. The quantitative relationship between the two kinds of size measure is further complicated by the fact that each system may have a different mix of technologies. One of the goals of software engineering is to provide development methods and tools which maximize the amount of functionality that a system can deliver to the user and help to reduce the amount of hand-crafted code as far as possible. The same is true for the activities of analysis and design. We intentionally weaken the dependency of effort on the functional size of a system, even in the course of a project, when we change to more efficient technologies. Therefore taking functional size as a base for estimating the effort is against our own intentions.

Of course, the situation is very different if one wants to compare alternate environments, process models and other parameters of software development. Functional size measured by function points seems to be ideally suited as a meaningful common reference. But this purpose of comparing is directed towards research and strategic studies, not towards the routine work of our company.

Estimating successfully with function points is feasible under very special circumstances. Given a stable process every measure for an upstream process entity can serve as an estimator for downstream entities like cost and schedule. All that is required is that the other parameters of the process are well known and that there are historical records available to support the estimating process. These conditions, however, are not easily met and are certainly not present in our company. One reason is the size of our projects, the other is the heavy investment in new technology and their subsequent effects which rendered our historical data almost useless.

We have now reached the point where our shift away from function points must be considered a logical (and psychological) consequence of learning about function points and about ourselves. This shift was a dangerous moment in the history of our metrics program [7]. It is reported that most metrics projects are abandoned after a few years. Luckily, our project could be restarted. The turn-around happened when we decided to change our approach totally [8].

4 The Processes of Project Planning and Tracking

This is the third group of findings and it resulted in accepting our own immaturity as a metrics project. This happened in a sequence of steps. In April 1999 we took over the following new strategy:

- We decided that estimation and controling were the most urgent techniques that needed improvement
- We developed a model which integrated project estimating, planning, tracking, and controling. Lots of data existed which did not fit to each other
- We decided to follow all the organizational rules that had been established for any proper software development project. Previously we had been a central function
- We would be a prototype for everything we were going to preach to others
- A steering committee was formed by leaders of software development projects, i.e. by experienced practitioners [9].

Two months later the next step was necessary to overcome the language barrier between the practitioners and the metrics project. We explicitly declared that we not only followed the organizational rules but considered ourselves just like another software development project. We adopted the process model of iterative software development, stressed the importance of architecture, components and reuse. Gathering information from the invaluable resources of the internet had previously appeared to be something like „doing research“ (and had been looked upon with suspicion). It could now be reframed as „looking for reusable components“.

We had not anticipated the most dramatic step which came next and was forced on us by the practitioners in the steering committe. With the slogan „doing to you what is done to us“ the committee confronted us with any deviation from our plans, which happened again and again and which we tried to excuse with „totally unforeseeable“ etc. – we had landed where most of the projects had gone before. That was the moment of truth. We recognized that we all had to learn some very hard lessons before estimation and controling could be improved to the level of professionalism which we had intended to reach in one giant stroke when making Function Point Analysis the corner stone of our metrics system.

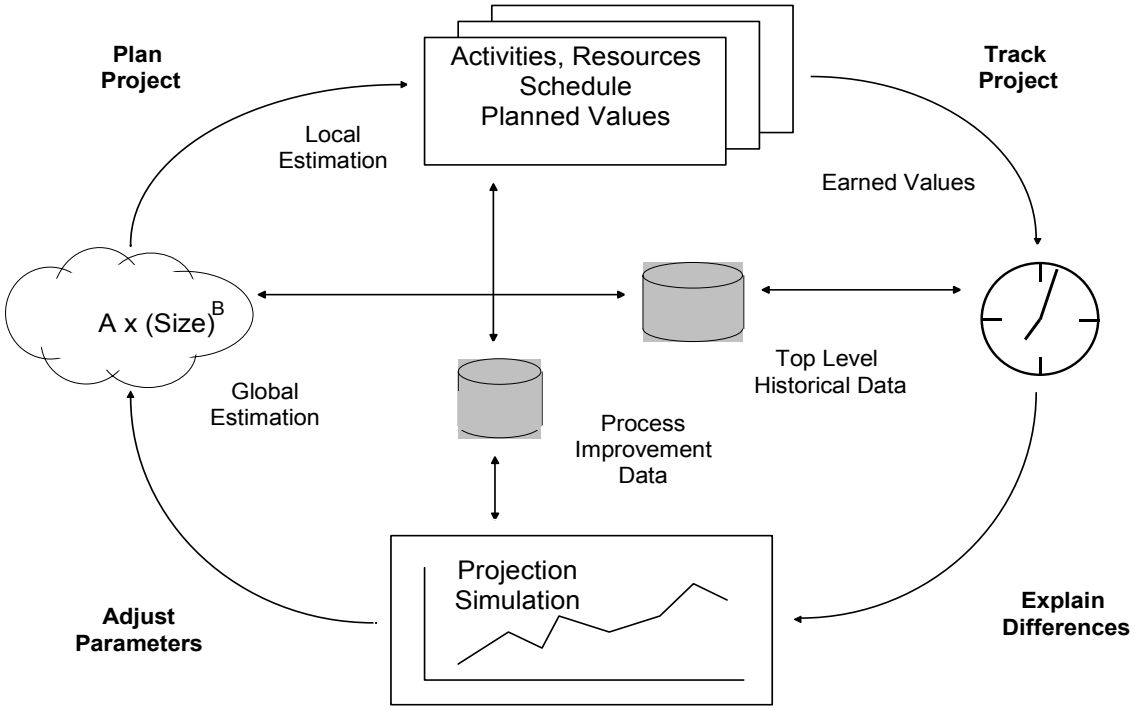


Figure 2: Our Integrated Model for the main activities we wanted to improve. This model convinced management and staff that the whole process was much more important than any single method of measurement could possibly be. Function points became a mere option.

Our goals are still to establish a business view upon our software development activities and controling our improvement efforts from a financial point of view. Considering the risks – what regularly happened to metrics projects and how we had gone astray – we are going to base the next project cycle on the Capability Maturity Model (CMM) of the Software Engineering Institue with its wealth of practical and successful experiences [10].

Summing up and Looking into the Future

Modeling the functional aspects of systems is a standard exercise in software development nowadays. Counting implies modeling and it doesn't seem to make sense to model the functional aspects of systems twice, once for development purposes and once for quantification. We would prefer to introduce FPA as an add-on to the mainstream of current modeling techniques, e.g. as the quantification of use cases and domain models when using UML.

The foundations of FPA are not well enough defined to tackle the interrelationships between functional aspects and the architecture of systems. A common complaint among our software engineers was the dependency of functional size on the cutting of a system.

We no longer believe that FPA is a suitable tool within an environment which is still immature and is looking for a path to higher process maturity. Declaring that there is a way to measure the value of a system independent of its technology, quality and development process – and failing to proof it - is contrary to building a project culture which is founded upon

gradual improvement of the given environment: beginning with sorting out what works and what doesn't, finding out how to do disciplined work, committing to realizeable plans, trying to integrate different process cultures into a common framework.

If the usefulness of FPA depends on the degree of process maturity in the organization, then there may be a place for FPA in a mature organization. When a high degree of maturity in planning, controling, risk management, and integration of software development is reached, then it could be justified to base business decisions on models incorporating FPs. Using FPs for business decisions requires at least a solid statistical base of historical records to judge the uncertainty of the effect of functional size on software size and effort under different circumstances. However, this is a static business model which in itself is limited in usefulness. We expect that dynamic process modeling will be the key to put software measurement in general into a fresh perspective [11].

References and Annotations

[1] Function Point Analysis was invented by Allan Albrecht in 1979. Its purpose is to quantify the functionality of software as seen by the the user. See the homepage of the International Function Point User Group (IFPUG) http://www.ifpug.org/home/docs/ifpughome.html. In the following discussion „function points" always refers to Unadjusted Function Points.

[2] Learning happens among people and we would like to express our thanks to all who set the stage and participated in our adventure. The measurement program in our company was initiated by Klaus Eigenwillig, who created our vision and changed the priorities and goals of our software and process engineering group from predominantly technological to financial aspects and to the performance of processes.

[3] We are very grateful to Robert Hürten, Dietmar Wuksch, Axel Fabry, and Manfred Bundschuh for sharing their knowledge and enthusiasm with us. They all had implemented FPA sucsessfully in other environments and continue to support us and shape our goals. DASMA can be reached at http://www.dasma.de/

[4] Notably the COSMIC Initiative (http://www.cosmicon.com/) is instrumental in creating a new generation of functional size measurement.

[5] The transaction functions (input, output, query) proved more useful than the data functions, because they did not interfere with modeling practices which heavily depended upon data models.

[6] http://www.psmsc.com/psm_doc.html

[7] Three people deserve the merits for keeping our project alive. Our manager Hans-Peter Müller never stopped fighting for a process focus on software development, his successor Karl-Heinz Thunemann put the project into a new organizational context, which turned out to be a very wise decision. Reinhard Terrahe funded the effort and committed himself to ist success.

[8] The new strategy of the project resulted from a joint effort of Renate Küntzel and the author. The project owes much to her insights into business processes, her energy and high professional standards in turning strategy into actions.

[9] We would like to thank Wilhelm Bitterberg, Horst Gerlach, Ernst Sackmann, Raimund Schulte, Johann Tamme, and Michael von Kopp Ostrowski for constantly charging the project for practical results. They offered hot discussions and emotional support as well.

[10] http://www.sei.cmu.edu/managing/managing.html

[11] See e.g. http://sunset.usc.edu/classes/cs599_99/spd/spd.html.

Getting Quick Wins with Software Metrics SLIM Estimate

Gren Bingham
QSM Metric Consult, The Netherlands, www.qsm.com

In the practice of software metrics and software process consulting, we often hear comments that can be summarized as "we don't have the time to make the changes to do things better and for lower cost." We have found that a good way of overcoming these objections is to give the organization "quick wins" from using software metrics. This helps to win the hearts and minds of the people doing the work.

In case after case we have shown that it is possible to get serious improvements in software planning very fast. To do this we concentrate first on giving reliable answers to the main questions that customers and managers want answered about software projects:

How long will this project take to complete?
How much will it cost?
How good will the software be when it is delivered?

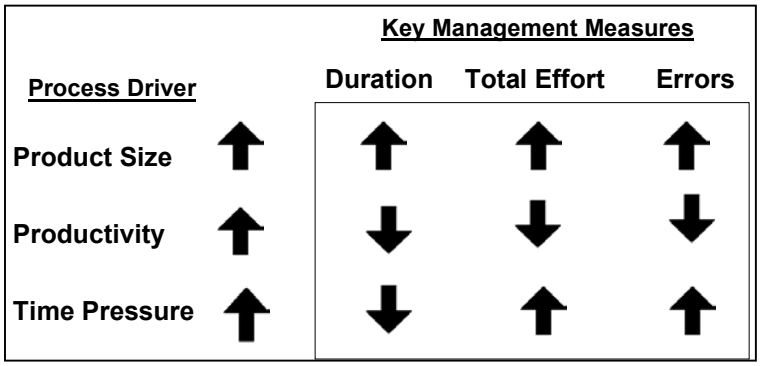
Surprisingly to many software managers these questions can be answered in reliable manner very easily. Within a very short period of making a commitment to improve the software process in an organization, much more accurate estimation and planning can be in place and working. The results are reliable and give estimates and plans that are tailored to the specific performance characteristics of the particular development organization. We have seen companies getting these good results in less than one month from starting.

The Concept

All of us who have been involved in software projects have seen that the following relationships hold true:

Relationship of Software Process Drivers to Key Management Measures

Tool Description



The above diagram shows the relationship of key management numbers to key aspects of software development, namely size, process productivity, and time pressure.

After analyzing thousands of software projects, Lawrence Putnam Sr., President of Quantitative Software Management, Inc., has been able to quantify these relationships and has described them in the QSM Software Process Equation:

PP = size / (effort/B)(1/3) * (time)**(4/3)**

We segment the observed domain of the productivity parameter (PP) into a 40-point management scale, the **QSM Productivity Index or "PI"**. ("B" is a special "skills factor" relating to the size of the product.) By putting in the size, time, and effort, from finished projects, into the software equation the process productivity parameter can be calculated. The PI offers a high level indication of the performance of the software development environment. It summarizes the impact of the many factors that influence software production by analyzing outcome measure from the projects. Improving the PI requires investment in the development process, and takes time.

Time and staffing strategies are important factors in that influence software development. If we look deeper into the project database, we can define a "staffing index" which reveals the "time pressure" under which projects are being developed. Increased time pressure is what happens when you develop the same functionality being developed in say 15 rather than 18 months. We can calculate the Manpower Buildup Parameter as follows:

MBP = total effort/(development time)3**

QSM segments the MBP domain into a 14-point management scale, which we call the **QSM Staff (or Manpower) Buildup Index or MBI**. Time pressure and the associated intense staffing strategies have a dramatic and negative impact on cost and quality of software developments. The time allowed for a project is, of course, a decision that can be changed in an instant by management fiat while planning the project. Thus, understanding the time-effort trade-off offers great scope for cost and quality management in software developments.

To estimate the expected time and effort for a new software project or an enhancement project, it is possible to select a set of relevant recent projects from the development organization, calculate their Productivity Indices and use these to select a representative PI parameter to apply in the estimation version of the software equation. By using available information on typical project times, this can generate an estimate and a plan based on typical

time pressure. We can then explore realistic staffing alternatives to meet other management constraints on the project, including product reliability requirements. We call these "viable" estimates because, being based on the way the organization works, they are realistic and achievable.

These basic concepts are at the heart of the QSM software project estimating tool SLIM Estimate. Those interested can get enough detail to build a spread sheet model by following Putnam's discussion in his first book, written with Ware Myers*, Measures for Excellence,* (L. H. Putnam and W. Myers, Prentice Hall, New York, 1991). Prudence makes us suggest that you not use the spreadsheet models for business critical estimations.

The SLIM Estimate Tool

Those who do the exercise of creating a simple spread sheet model using the software equation quickly appreciate the benefits and sophistication of the estimation tool that Putnam and his team have created to deploy the relationships he has defined. It is called SLIM Estimate. SLIM Estimate has proved its strengths and value in the market for over many years, and is now in its fourth major release.

SLIM Estimate generates software project time and effort estimates, plans, and milestones. These are presented in a "probabilistic" way that allows good insight into the risks implicit in the project estimate and project plan. This means that you have the possibility to offer the customer a schedule and cost which has, say, a 95% certainty of being correct, while using numbers to plan the project, internally to the development group, which are at the 50% confidence level (i.e. the "typical" or most-likely performance)

SLIM Estimate can be used "right from the box" because it includes information derived from over 5000 projects in the QSM database. This allows you to generate estimates that reflect typical behavior of software developers executing similar projects. However, by using the historical project data from your own organization you can tailor the results to reflect the way your organization develops software.

SLIM Estimate allows the user to maintain historical databases and evaluate them in sufficient detail to provide tuning factors and parameters that tailor estimations to match the environment from which the history comes.

Using historical data to tune the estimation pays off. A classic test of estimation engines is to "play back" a completed project. That is to say, based on the size predict the time and effort using all the facilities of the tool. SLIM Estimate can replay a project to within one per-cent of actual time and effort, when in the hands of an experienced user. This suggests that if you have a reasonable number of relevant projects, you can get very, very accurate estimations using SLIM Estimate.

In addition to time and effort estimation, there is a sophisticated error model incorporated into the tools that predicts the mean time to defect. It also forecasts the number of defect to be discovered during the development life-cycle and the distribution of those defects over time.

If you have historical error data, the model can be tuned to reflect the quality characteristics of your development organization.

One of the most important features of SLIM Estimate is the ease with which a variety of optional project development scenarios can be created, assessed, and saved. The tools is critical in doing this because the trade off between time and effort in software development in not linear. To state the obvious, this means that if you cut the time for a software project in half, you cannot accomplish it with twice the number of staff. SLIM Estimate has the size-time-effort-quality trade-off built in. This allows you to explore the impact of changing time, staffing strategies, and functional content on the estimate and the project plan. You can store the most interesting scenarios in the log, possibly for later discussion with your customer or management team.

Finally, SLIM Estimate comes with an excellent reporting facility which can be used directly for making presentations of the results using a PC, as well as creating printed reports.

SLIM Estimate is one of four tools that Quantitative Software Management provides to support achieving quick wins in software process improvement. The other three management tools are:

SLIM Control, which tracks project progress and dynamically replans the project in response to changing circumstances (such as size growth, staff changes, and so on);

SLIM Metrics, which is a metrics repository and analytic engine supporting the analysis of an essentially unlimited number of process variables;

SLIM Master Plan, which allows clear visualization, integration, and easy management of plans for portfolios of software projects and software releases.

The author's e-mail address is: gren_bingham@qsm.com. and he will be happy to respond to your questions. The QSM web site can be found at the URL: http://www.qsm.com. Gren Bingham runs the Amsterdam office of Quantitative Software Management and handles QSM business in a variety of European countries.

END OF PROPOSED ARTICLE. THE INFORMATION BELOW IS FOR BACKGROUND OR FILLER IF YOU WISH TO USE ANY OF IT.

Grenville Bingham has been active in software process improvement since 1986, when he created the Productivity Enhancement Program, a subscription service supplied to over 100 leading European and South African organizations, for most of its life by Computer Sciences Corporation.

Gren. has been directly associated with Quantitative Software Management since 1988 and is in charge of their Amsterdam office serving European QSM customers.

After completing university, Gren started his career in software development and consulting as a systems software developer with IBM in the USA.

Tool Description

Short Introductory Overview of QSM, the Company, plus its Products and Concepts
Quantitative Software Management, Inc.

About QSM

FOUNDED IN 1978, QUANTITATIVE SOFTWARE MANAGEMENT, INC. (QSM), IS THE MOST SENIOR SOFTWARE MANAGEMENT FIRM IN THE INDUSTRY. QSM METHODS, TOOLS, AND TRAINING PROVIDE PRO-ACTIVE SOLUTIONS FOR SOFTWARE PRODUCTIVITY MEASUREMENT AND IMPROVEMENT, COST AND SCHEDULE ESTIMATING, SIZE ESTIMATING, AND *RUNAWAY PROJECT PREVENTION*. THEY ARE USED WORLDWIDE BY FORTUNE 500 CLASS COMPANIES AND BY GOVERNMENT AGENCIES. THE QSM PROCESS APPROACH IS USED TO MEASURE ALL TYPES OF SOFTWARE DEVELOPMENT, FROM DETAILED MICRO CODE, REAL-TIME AVIONICS AND WEAPONS SYSTEMS, PROCESS CONTROL, AND SYSTEMS SOFTWARE TO LARGE FINANCIAL, BANKING AND MIS SYSTEMS.

QSM GROWS AND MAINTAINS THE WORLD'S LARGEST SOFTWARE DEVELOPMENT PROJECT HISTORY DATABASE. THE DATABASE CAPTURES STATISTICS ON SIZE, PRODUCTIVITY, TIME, EFFORT, COST, STAFFING, RELIABILITY, AND NUMEROUS PRODUCT, ENVIRONMENTAL, AND PROCESS-RELATED ATTRIBUTES. THE DATABASE CURRENTLY CONTAINS OVER 5,000 PROJECTS WITH STATISTICAL SIGNIFICANCE IN ALL APPLICATION DOMAINS. IT IS GROWING AT A RATE OF APPROXIMATELY 250 PROJECTS PER YEAR. MANY OF THE WORLD'S MAJOR SOFTWARE PRODUCERS ARE REPRESENTED IN THE DATABASE.

QSM OPERATES OFFICES IN WASHINGTON, DC; PITTSFIELD, MA; PHOENIX, AZ; LONDON, UK; AMSTERDAM, NETHERLANDS; AND PARIS, FRANCE. QSM'S FOUNDER AND PRESIDENT, LAWRENCE H. PUTNAM, SR. IS A WORLD-RENOWNED AUTHOR, LECTURER, AND CONSULTANT ON SOFTWARE PRODUCTIVITY, QUALITY, AND LIFE CYCLE ESTIMATING. HE HAS WRITTEN OVER 30 TECHNICAL PAPERS AND FOUR BOOKS ON THE SUBJECT. THE BOOKS ARE:

- *Software Cost Estimating and Life Cycle Control, Getting the Management Numbers*, published by the IEEE Computer Society
- *Measures for Excellence: Reliable Software On Time, Within Budget*, published by Prentice Hall
- *Executive Briefing: Controlling Software Development*, published by the IEEE Computer Society Press
- *Industrial Strength Software: Effective Management Using Measurement*, published by the IEEE Computer Society Press

QSM Services and Products

COMMERCIAL SERVICES

- Basic Measures / Competitive Benchmarking
- Software Project Office Startup and Support
- Independent Project Estimates
- Runaway Project Replanning Assessments

Tool Description

- Monthly Project Health Check Assessments
- Competitive Bid Assessments
- Make vs. Buy Economic Analysis
- Research Studies
- Software Product Valuations
- Expert Witness in Support of Contested Contracts

COMMERCIAL PRODUCTS

- ***SLIM-Estimate*** - A PC-hosted project estimation and planning tool which implements QSM's Software Lifecycle Management (SLIM) method for determining the amount of function required to satisfy a given set of software requirements and then allows managers to identify the best strategy for building a corresponding product, shortening cycle time, reducing cost, improving quality, and minimizing risk.

- ***SLIM-Control*** - A PC-hosted project tracking and oversight tool which implements Statistical Process Control techniques for assessing the status (plan versus actual with forecast to completion) of built-in and user-defined measures and metrics.

- ***SLIM-Metrics*** - A PC-hosted metrics repository tool which allows the user to assess competitive position, identify development bottlenecks, quantify the benefits of improvement, and support planning for future projects.

- ***SLIM-MasterPlan*** – A PC-hosted analysis tool which accepts time-series measurement data from multiple sources and provides a single-chart portrayal of each measurement, either by individual source or in aggregate.

TRAINING SLIM-ESTIMATE AND SLIM-CONTROL

- SLIM-Metrics

QSM Process and Tools

Figure 1 illustrates the Software Lifecycle Management (SLIM) process and shows how the QSM tools work together to support that process.

QSM Metrics, Methodology, and Key Concepts

SOFTWARE, TODAY, IS THE KEY ELEMENT IN SYSTEMS SUPPORTING THE BULK OF COMMERCE THROUGHOUT THE WORLD. HOW FAST SUCH SYSTEMS GET INTO SERVICE AND HOW GOOD THEY ARE HAS LARGE ECONOMIC SIGNIFICANCE. THE KEY TO REDUCING CYCLE TIME OR TIME TO MARKET AND ENSURING QUALITY IS COGNIZANCE OF A FEW CRITICAL MANAGEMENT NUMBERS AND THEIR RELATIONSHIPS.

Software Production Relationship

QUANTITATIVE SOFTWARE MANAGEMENT, INC. DEVELOPED A SOFTWARE PRODUCTION EQUATION THAT DESCRIBES THE RELATIONSHIP BETWEEN THESE MANAGEMENT NUMBERS.

Tool Description

THIS EQUATION HAS BEEN VALIDATED OVER THE LAST 15 YEARS WITH REAL DATA FROM MANY THOUSANDS OF COMPLETED PROJECTS REPRESENTING EVERY CONCEIVABLE APPLICATION TYPE. THE EQUATION TAKES THE FOLLOWING CONCEPTUAL FORM:

Quantity of Function Created = Process Productivity × Effort × Schedule

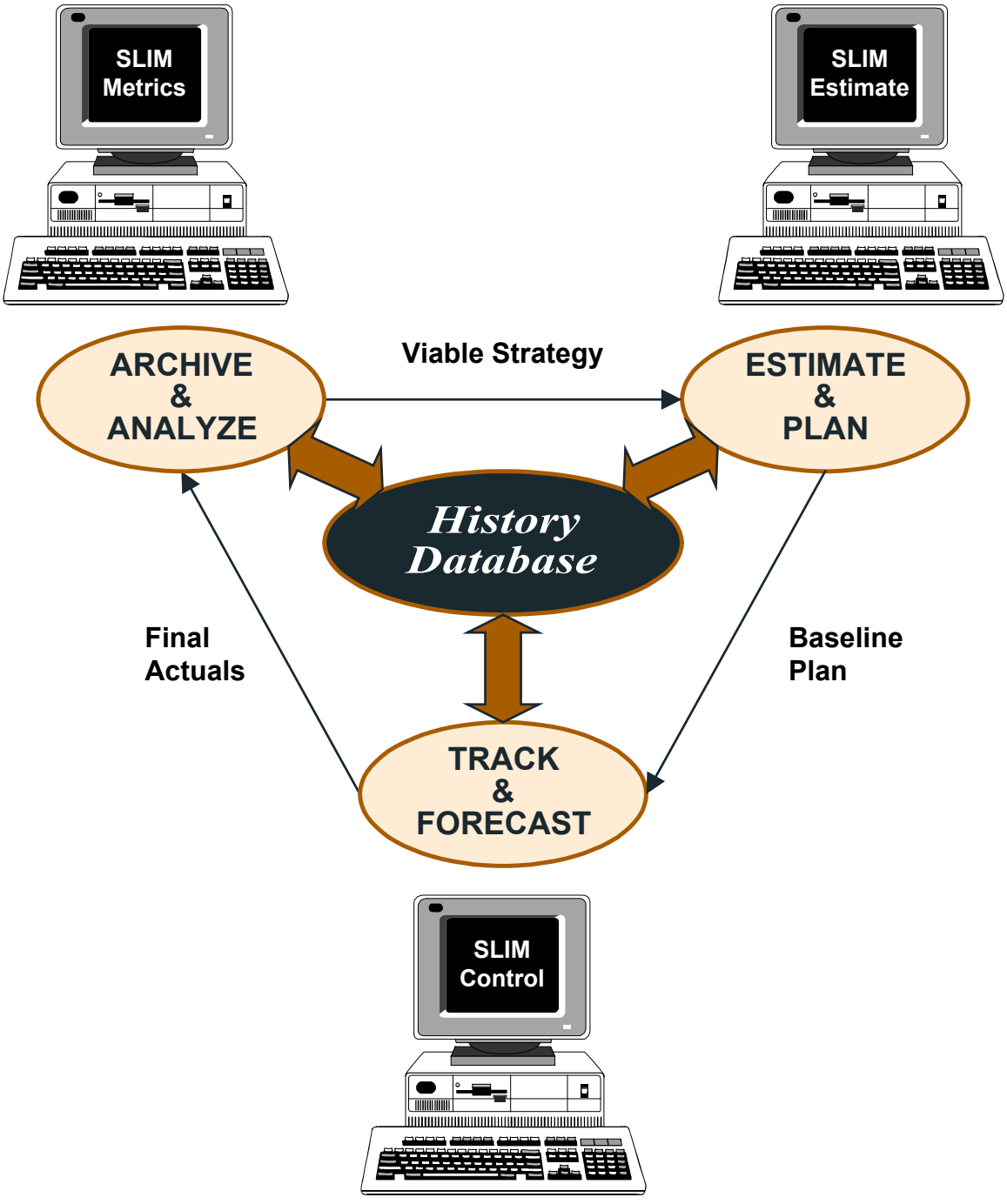


Figure 1: *QSM Process and Tools*

Tool Description

THIS SAYS THAT THE PRODUCT OF THE TIME AND EFFORT COUPLED WITH THE PROCESS PRODUCTIVITY OF THE DEVELOPMENT ORGANIZATION DETERMINES HOW MUCH FUNCTIONALITY CAN BE DELIVERED. EXTENSIVE EMPIRICAL STUDY OF SOFTWARE DATA HAS SHOWN THAT VERY STRONG NON-LINEARITIES EXIST IN THE ACTUAL RELATIONSHIP BETWEEN THESE PARAMETERS. THE COMPUTATIONAL FORM OF THE SOFTWARE PRODUCTION EQUATION ACCOUNTS FOR THESE NON-LINEARITIES IN THE FORM OF EXPONENTS APPLIED TO SOME OF THE PARAMETERS:

$$S = C_k \times \left(\frac{E_d}{\beta} \right)^{1/3} \times t_d^{4/3}$$

where

S (Size) is the quantity of developed functionality measured in effective source lines of code (ESLOC). Note that other measures of functionality could also be used given an appropriate gearing factor.

C_k (Process Productivity Parameter) is the organization's development process efficiency. The value of this parameter is determined by the organization's historic data.

E_d (Main Build Effort) is the amount of labor, measured in person-years, required to complete all the activities included in the Software Main Build (typically software detail design, coding, and integration to the point where all the operational capability exists and 95% of the total defects have been found and fixed).

β (Special Skills Factor) provides for specialized integration, testing, documentation, and management skills that come into play as system size increases. Essentially, it is a complexity adjustment factor for size.

t_d (Main Build Time) is the amount of calendar time in years required to complete all the activities included in the Software Main Build (see Main Build Effort above).

IF WE RE-ARRANGE THE EQUATION TO ISOLATE EFFORT, AND THEN MULTIPLY BY A FULLY BURDENED LABOR RATE, WE OBTAIN A SOFTWARE PRODUCTION EQUATION THAT LOOKS LIKE THIS:

$$MB_{cost} = \left(\frac{S}{C_k} \right)^3 \times \frac{1}{t_d^4} \times \beta \times L$$

where

MB_{cost} (Main Build Cost) is the amount of money in $ required to complete all the activities included in the Software Main Build (see Main Build Effort above).

L (Burdened Labor Rate) is the organization's average fully burdened labor rate in $/person-year.

THIS EQUATION HAS ENORMOUS ECONOMIC LEVERAGE THAT CAN BE EXPLOITED BY MANAGEMENT. REDUCING SIZE (DEFERRING FUNCTIONALITY OR EMPLOYING REUSE) CAUSES THE COST TO GO DOWN AS THE CUBE OF THE SIZE REDUCTION. IMPROVING PROCESS EFFICIENCY (INVESTMENT IN METHODS, TOOLS, TRAINING) CAUSES THE COST TO GO DOWN AS THE CUBE OF THE INCREASE IN THE PROCESS PRODUCTIVITY PARAMETER. LENGTHENING THE TIME (NEGOTIATING A MORE RELAXED SCHEDULE WITH THE CUSTOMER) CAUSES THE COST TO GO DOWN AS THE FOURTH POWER OF THE SCHEDULE EXTENSION. DURING A GIVEN DEVELOPMENT PROJECT ONE, TWO, OR SOMETIMES ALL THREE OF THESE SITUATIONS CAN BE MADE TO HAPPEN BY MANAGEMENT.

THE NATURE OF THE SOFTWARE PRODUCTION EQUATION AND ITS CONSTITUENT UNITS CAUSES THE RANGE OF POSSIBLE PROCESS PRODUCTIVITY PARAMETER VALUES TO BE ENORMOUS. IN AN ATTEMPT TO SIMPLIFY THIS TERM, QSM HAS MAPPED THE PROCESS PRODUCTIVITY PARAMETER RANGE ONTO A SCALE THAT RANGES FROM 1 TO 40 AND CALLS VALUES ON THIS SCALE PRODUCTIVITY INDICES (PI). ALL SOFTWARE PROJECTS ENCOUNTERED SO FAR BY QSM HAVE FIT WITHIN THIS RANGE.

FOR A MORE DETAILED DEVELOPMENT OF THE SOFTWARE PRODUCTION RELATIONSHIP AND THE PRODUCTIVITY INDEX, SEE LAWRENCE H. PUTNAM AND WARE MYERS, *MEASURES FOR EXCELLENCE: RELIABLE SOFTWARE ON TIME, WITHIN BUDGET*, YOURDON PRESS, NEW YORK, 1982, 284 PP.

Manpower Buildup Relationship

THE SPEED AT WHICH MANAGEMENT APPLIES PEOPLE TO A SOFTWARE PROJECT HAS A LARGE IMPACT ON COST AND QUALITY WITH ONLY A MODEST IMPACT ON SCHEDULE. QSM HAS DEVELOPED A MEASURE OF THIS STAFFING BEHAVIOR PHENOMENON CALLED A MANPOWER BUILDUP PARAMETER AND HAS DEFINED THE VALUE OF THIS PARAMETER IN TERMS OF EFFORT AND SCHEDULE WITH THE FOLLOWING MANPOWER BUILDUP EQUATION:

$$g = \frac{E_d}{\beta \times t_d^3}$$

where

g (Manpower Buildup Parameter) is the organization's development process efficiency. The value of this parameter is determined by the organization's historic data.

E_d (Main Build Effort) is the amount of labor, measured in person-years, required to complete all the activities included in the Software Main Build (typically software detail design, coding, and integration to the point where all the operational capability exists and 95% of the total defects have been found and fixed).

β (Special Skills Factor) provides for specialized integration, testing, documentation, and management skills that come into play as system size increases. Essentially, it is a complexity adjustment factor for size.

t_d (Main Build Time) is the amount of calendar time in years required to complete all the activities included in the Software Main Build (see Main Build Effort above).

THE NATURE OF THE MANPOWER BUILDUP EQUATION AND ITS CONSTITUENT UNITS CAUSES THE RANGE OF POSSIBLE MANPOWER BUILDUP PARAMETER VALUES TO BE ENORMOUS. IN AN ATTEMPT TO SIMPLIFY THIS TERM, QSM HAS MAPPED THE MANPOWER BUILDUP PARAMETER RANGE ONTO A SCALE THAT RANGES FROM -6 TO +6 AND CALLS VALUES ON THIS SCALE MANPOWER BUILDUP INDICES (MBI).

FOR A MORE DETAILED DEVELOPMENT OF THE MANPOWER BUILDUP RELATIONSHIP AND THE MANPOWER BUILDUP INDEX, SEE LAWRENCE H. PUTNAM AND WARE MYERS, *MEASURES FOR EXCELLENCE: RELIABLE SOFTWARE ON TIME, WITHIN BUDGET*, YOURDON PRESS, NEW YORK, 1982, 284 PP.

Process Improvement

THE PROCESS PRODUCTIVITY PARAMETER IN THE SOFTWARE PRODUCTION EQUATION INCREASES AS THE ORGANIZATION BECOMES MORE PROCESS EFFICIENT. BY PROCESS EFFICIENCY WE MEAN THE LONG TERM, CONTINUOUS IMPROVEMENT NOTION THE JAPANESE DESCRIBE AS *KAIZEN*.

CAPITAL INVESTMENT

THE PROCESS PRODUCTIVITY PARAMETER IS THE CAPITAL INVESTMENT TERM IN THE SOFTWARE PRODUCTION EQUATION. *IT COSTS MONEY, AND IT TAKES TIME AND DISCIPLINE TO MAKE PROCESS IMPROVEMENT HAPPEN.* THE PAYOFF IS HANDSOME FOR THOSE WILLING TO MAKE THE INVESTMENT.

IN THE EARLY DAYS OF SOFTWARE DEVELOPMENT, PROCESS IMPROVEMENT WAS REALIZED PRIMARILY BY REMOVING BOTTLENECKS AND REDUCING QUEUE TIME. THE MEASURED PAYOFF FOR THESE IMPROVEMENTS WAS CONSIDERABLE.

NOW, AS IT WILL BE TO A GREATER DEGREE IN THE FUTURE, PROCESS IMPROVEMENT IS MORE DIFFICULT TO MAKE HAPPEN BECAUSE IT INVOLVES CHANGING THE WAY ORGANIZATIONS WORK RATHER THAN JUST REMOVING BOTTLENECKS. SIGNIFICANT CULTURE CHANGE IS NECESSARY. CONSIDERABLE RESISTANCE IS THE NORM; HOWEVER, THE PAIN IN MAKING IT HAPPEN IS WELL WORTH THE SUFFERING. PROCESS IMPROVEMENT RESULTS IN LOWER DEVELOPMENT COST, REDUCED CYCLE TIME, AND IMPROVED PRODUCT QUALITY. A FEW GOOD COMPANIES HAVE BEEN DOING IT FOR YEARS AND THEIR RESULTS SHOW THAT IT PAYS OFF WITH ROIS IN THE RANGE OF 70% TO 100% PER YEAR.

SEI CMM LINKAGE WITH THE QSM PRODUCTIVITY INDEX

QSM HAS BEEN ABLE TO ESTABLISH A MAPPING BETWEEN ITS PRODUCTIVITY INDEX AND THE SOFTWARE ENGINEERING INSTITUTE'S CAPABILITY MATURITY MODEL MATURITY LEVELS. FOR A DESCRIPTION OF THE BASIS FOR THIS MAPPING, SEE LAWRENCE H. PUTNAM, *LINKING THE QSM PRODUCTIVITY INDEX WITH THE SEI MATURITY LEVEL AND PROJECTING TRANSITIONS TO VARIOUS MATURITY LEVELS*, MCLEAN, VA, FEBRUARY 1992.

PROCESS IMPROVEMENT SUMMARY

LARGE ECONOMIC LEVERAGE IS POSSIBLE FROM PROCESS IMPROVEMENT. IN FACT, INVESTING IN PROCESS IMPROVEMENT CAN, DURING THE FIRST GENERATION OF PROJECTS, SAVE ENOUGH MONEY TO PAY BACK THE INVESTMENT. HISTORIC DATA IN THE QSM DATABASE DEMONSTRATES THAT REAL COMPANIES HAVE DONE IT AND THE SAVINGS DO HAPPEN. WHAT IS REQUIRED TO MAKE IT HAPPEN?

- Taking process improvement seriously.
- Making the necessary investments.
- Getting full management support behind the process improvement program.
- Working hard on continuous improvement (*Kaizen* principle)

Buglione, L.: Misurare il Software. Quantità, qualità, standards e miglioramento di processo nell'Information Technology

Franco Angeli, Collana "Informatica & Organizzazioni" (724.20), 1999 (239 pp.)
ISBN 88-464-1729-1

EXCERPT OF THE PREFACE (*by Alain Abran*):

Measurement is so much part of our daily life at the beginning of this third millenium that we take most of it for granted and do not fully appreciate the challenges hidden behind such an apparently straightforward concept. The measures we use regularly may look simple, however, for many of them it took mankind centuries, if not longer, to develop and refine the measurement instruments, quantitatively and with consensus, the multiple representations of a single concept in an enormous variety of contexts, and also to arrive at the determination of what needs to be measured.

In the emerging field of software-related technologies, most of the measurement lessons from the past are not being learned and patience is not on the agenda. Expectations are high, however, and it is assumed that even challenging concepts, such as software quality, for which no consensus has yet been achieved can be easily measured, that measurement instruments can be designed rapidly and that the measurement process costs will be close to nil.

This book is aimed at contributing to the development of measures and measurement instruments in the software technology field. While addressing the highly discussed and visible issue of the measurement of software quality, the work looks far beyond the current literature on the as yet poorly defined domain of software measurement and brings to it the wisdom of centuries of work on the geometrical representation of simultaneous dimensions.

Tool Description

This open model, called **QEST** (**Q**uality factor + **E**conomic, **S**ocial & **T**echnical dimensions), has been developed to handle, simultaneously and concurrently, three-dimensional perspectives of performance.

Key features of the QEST model incorporated into the LIME model are:

- a 3D geometrical construction and representation;
- integrated quantitative and qualitative evaluation from the three concurrent viewpoints: management (*economic viewpoint*), user (*social viewpoint*) and technical personnel (*technical viewpoint*);
- use of standards such as ISO/IEC 9126 standard on Software Product Quality and Function Point Analysis.

This approach is quite innovative in this emerging field and careful study could pave the way for a new approach and an improved level of sophistication in the measurement of software.

Further information (in Italian) available at: http://www.francoangeli.it/libri/724000020.htm

Dumke, R.; Abran, A.: Software Measurement – Current Trends in Research and Practice

DUV Publisher, Wiesbaden, 1999 (269 p.) *ISBN 3-8244-6876-X*

This new book includes key papers presented at the 8th International Workshop on Software Metrics in Magdeburg (Germany), September 1998. It is a collection of theoretical studies in the field of software measurement as well as experience reports on the application of software metrics in USA, Canadian, Netherlands, Belgian, France, England and German companies and universities. Some of these papers and reports describe new software measurement applications and paradigms for knowledge-based techniques, test service evaluation, factor analysis discussions and neural-fuzzy applications. Other address the multimedia systems and discuss the application of the Function Point approach for real-time systems, the evaluation of Y2K metrics, or they include experience reports about the implementation of measurement programs in industrial environments.

Poels, G.: On the Formal Aspects of the Measurement of Object-Oriented Software Specification

University of Leuven (Belgium), 1999 (507 p.)

This work contributes to the state-of-the-art in software measurement by proposing a suite of measures for object-oriented software specifications related to a particular layer in a software system's architecture: the enterprise model. This model of business entities, business events, and business rules builds the nucleus of a software system. Modern software development strategies, such as the one recommended by the MERODE research group at the K.U.Leuven's

Tool Description

Department of Applied Economics, start building a software system from the stable foundations laid by the enterprise model. As a consequence, measuring the enterprise model and its components offers great potential for the early assessment, prediction and control of the software engineering and management variables of interest. These include the cost of software development and maintenance, the quality of the software system, and the effectiveness and efficiency of the methods, techniques and tools that are used. However, this is not the only motivation for measurement. As a representation of the business within the scope of the software system, the enterprise model might tell us something about the business functioning itself. It is therefore a worthwhile exercise to conceptualise and quantify its attributes.

CSMR'2000:

4th Euromicro Working Conference on Software Maintenance and Reengineering,
Feb 29 - March 3, 2000, Zuerich, Switzerland
see: **http://www.uni-koblenz.de/~ist/CSMR200/**

IFPUG 2000, Spring:

International Function Point User Group Spring Conference,
April 2 - 4 , 2000, Jacksonville, Florida, USA
see: **http://www.ifpug.org/conferences/conf.html**

SQM 2000:

5. Kongress Software-Qualitätssicherung
3 - 4 April 2000, Beethovenhalle in Bonn, Germany
see: **http://www.sqs.de/**

Escom-Scope 2000:

The Science and Practice of Software Metrics
18 - 20 April, 2000, Munich, Germany
see: **http://www.escom.co.uk**

QW'2000:

Quality Week 2000,
May 30 - June 2, San Francisco, USA,
see: **http://www.soft.com/QualWeek/QW2K**

QAOOSE 2000:

Workshop on Quantitative Approaches in Object-Oriented Software Engineering
June 13, Sophia Antipolis and Cannes, France,
see: **http://www.iro.umontreal.ca/~sahraouh/qaoose/**

PROFES'2000:

International Conference on Product Focused Software Process Improvement
June 20-22, 2000, Oulu, Finland
see: **http://www.ele.vtt.fi/profes99/**

ICSR6:

Sixt International Conference on Software Reuse,
June 27-29, Vienna, Austria,
see: **http://www.spe.ucalgary.ca/icsr6/**

ICRS 2000:

5th International Conference on Reliable Software Technologies,
June 26 - 30, Potsdam, Germany
see: **http://www.ada-europe.org/conference2000.html**

FMSP 2000:

International Symposium on Software Testing and Analysis,
August 21 - 24, 2000, Portland, Oregon, USA
see: **http://www.ics.uci.edu/issta-fmsp**

IFPUG 2000, Fall:

International Function Point User Group Fall Conference,
September 11-15 , 2000, San Diego, USA
see**: http://www.ifpug.org/conferences/conf.html**

CONQUEST 2000:

Conference on Quality Engineering in Software Technology
September 14 - 15, 2000, Nuremberg, Germany
see: http://www.asqf.de/

UML 2000:

Third International Conference on the Unified Modeling Language
October 2 - 4, 2000, York, UK
see: http://www.cs.york.ac.uk/uml2000/

FESMA 2000:

3rd Conference on European Federation of Software Metrics Associations
October 2-6, 2000, Madrid, Spain
see: **http://www.fesma.org**

IWSM'2000:

10th International Workshop on Software Measurement
October 4 - 6, 2000, Berlin, Germany
see: **http://ivs.cs.uni-magdeburg.de/sw-eng/us/**

ISSRE 2000:

Eleven International Symposium on Software Reliability Engineering
October 8-11, 2000, San Jose, California
see: http://www.rstcorp.com/conferences/issre2000

ICSM 2000:

International Symposium on Software Maintenance
October 11-14, 2000, San Jose, California
see: **http://www.rstcorp.com/conferences/icsm2000**

PNSQC 2000:

2000 Pacific Northwest Software Quality Conference
October 16-18, 2000, Portland, Oregon
see: **http://www.pnsqc.org**

EuroSTAR 2000:

8th European International Conference on Software Testing Analysis & Review,
December 4 - 8, 2000, Copenhagen, Denmark
see: **http://www.eurostar.ie/**

METRICS 2001 & ESCOM 2001:

The Science and Practice of Software Metrics
April 2 - 6, 2000, London, England
see: **http://www.telecom.lth.se/**

Tool Description

see also: **OOIS, ECOOP** and **ESEC** European Conferences

Other Information Sources and Related Topics

- **http://rbse.jsc.nasa.gov/virt-lib/soft-eng.html**
  Software Engineering Virtual Library in Houston

- **http://www.mccabe.com/**
  McCabe & Associates. Commercial site offering products and services for software developers (i. e. Y2K, Testing or Quality Assurance)

- **http://www.sei.cmu.edu/**
  Software Engineering Institute of the U. S. Department of Defence at Carnegie Mellon University. Main objective of the Institute is to identify and promote successful software development practices.
  Exhaustive list of publications available for download.

- **http://dxsting.cern.ch/sting/sting.html**
  Software Technology INterest Group at CERN: their WEB-service is currently limited (due to "various reconfigurations") to a list of links to other information sources.

- **http://www.spr.com/index.htm**
  Software Productivity Research, Capers Jones. A commercial site offering products and services mainly for software estimation and planning.

- **http://fdd.gsfc.nasa.gov/seltext.html**
  The Software Engineering Laboratory at NASA/Goddard Space Flight Center. Some documents on software product and process improvements and findings from studies are available for download.

- **http://www.qucis.queensu.ca/Software-Engineering/**
  This site hosts the World-Wide Web archives for the USENET usegroup comp.software-eng. Some links to other information sources are also provided.

- **http://www.esi.es/**
  The European Software Institute,Spain

- **http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html**
  Software Engineering Management Research Laboratory at the University of Quebec, Montreal. Site offers research reports for download. One key focus area is the analysis and extension of the Function Point method.

- **http://www.SoftwareMetrics.com/**

Homepage of Longstreet Consulting. Offers products and services and some general information on Function Point Analysis.

- **http://www.utexas.edu/coe/sqi/**
  Software Quality Institute at the University of Texas at Austin. Offers comprehensive general information sources on software quality issues.

- **http://wwwtrese.cs.utwente.nl/~vdberg/thesis.htm**
  Klaas van den Berg: Software Measurement and Functional Programming (PhD thesis)

- **http://divcom.otago.ac.nz:800/com/infosci/smrl/home.htm**
  The Software Metrics Research Laboratory at the University of Otago (New Zealand).

- **http://ivs.cs.uni-magdeburg.de/sw-eng/us/**
  Homepage of the Software Measurement Laboratory at the University of Magdeburg.

- **http://www.cs.tu-berlin.de/~zuse/**
  Homepage of Dr. Horst Zuse

- **http://dec.bournemouth.ac.uk/ESERG/bibliography.html**
  Annotaded Bibliography on Object-Oriented Metrics

- **http://www.iso.ch/9000e/forum.html**
  The ISO 9000 Forum aims to facilitate communication between newcomers to Quality Management and those who, having already made the journey have experience to draw on and advice to share.

- **http://www.qa-inc.com/**
  Quality America, Inc's Home Page offers tools and services for quality improvement. Some articles for download are available.

- **http://www.quality.org/qc/**
  Exhaustive set of online quality resources, not limited to software quality issues

- **http://freedom.larc.nasa.gov/spqr/spqr.html**
  Software Productivity, Quality, and Reliability N-Team

- **http://www.qsm.com/**
  Homepage of the Quantitative Software Management (QSM) in the Netherlands

- **http://www.iese.fhg.de/**
  Homepage of the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany

- **http://www.highq.be/quality/besma.htm**
  Homepage of the Belgian Software Metrics Association (BeSMA) in Keebergen, Belgium

- **http://www.cetus-links.org/oo_metrics.html**
  Homepage of Manfred Schneider on Objects and Components

- **http://dec.bournemouth.ac.uk/ESERG/bibliography.html**
  An annotated bibliography of object-oriented metrics of the Empirical Software Engineering Research Group (ESERG) of the Bournemouth University, UK

News Groups

- **news:comp.software-eng**
- **news:comp.software.testing**
- **news:comp.software.measurement**

METRICS NEWS

VOLUME 4 *1999* *NUMBER 2*

CONTENTS

Call for Papers **3**

Conference Reports **9**

Announcements **39**

Position Papers **41**

Schmietendorf, A.; Scholz, A.:
The Performance Engineering Maturity Model **41**

Bierfert, H.:
Beginning a Measurement Program - A Learning Experience with Function Point Analysis **52**

Tool Descriptions **61**

New Books on Software Metrics **73**

Conferences Addressing Metrics Issues **75**

Metrics in the World-Wide Web **79**

ISSN 1431-8008