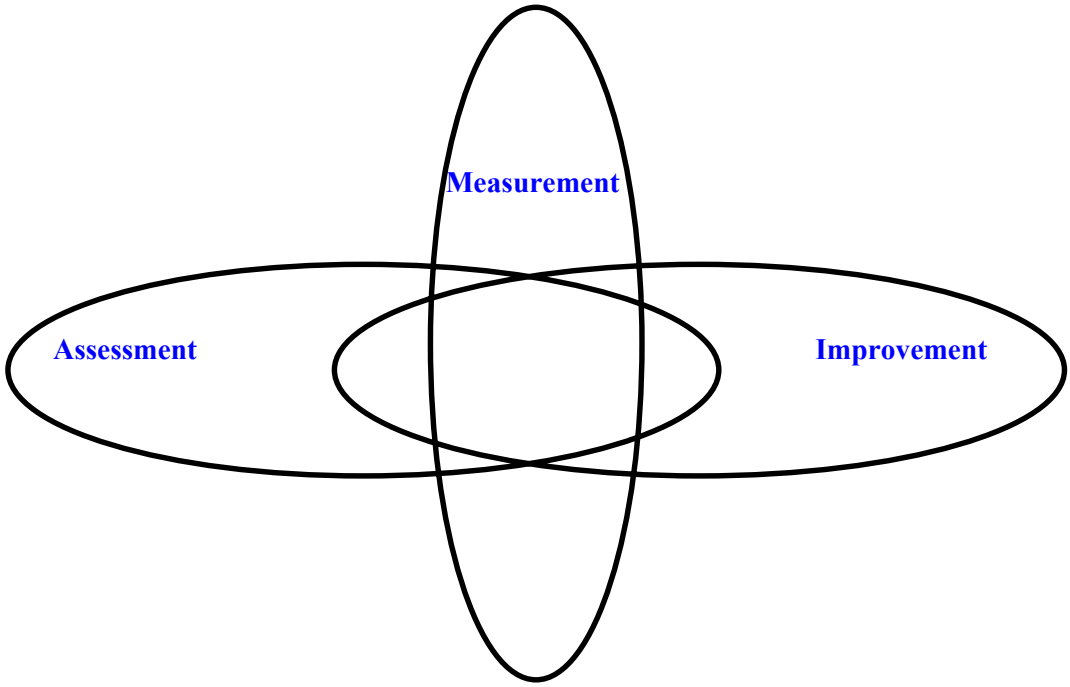
Volume 2, Number 1, June 1997

METRICS NEWS

Journal of the GI-Interest Group on Software Metrics



Editors: *R. Dumke, C. Ebert, E. Rudolph, H. Zuse*

Otto-von-Guericke-Universität

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

Editors:

Reiner Dumke
Professor on Software Engineering,
University of Magdeburg, Faculty of Informatics,
Postfach 4120, D-39016 Magdeburg, Germany
Tel.: +49-391-67-18664, Fax: +49-67-12810
email: dumke@irb.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.be

Eberhard Rudolph
Professor on Software Engineering
Hochschule Bremerhaven, FB2 System Analysis,
Mozartstr. 37, D-27570 Bremerhaven, Germany
Tel.: +49-471-26142, Fax: +49-471-207389
email: rudolph@oscar-e.hs-bremerhaven.de

Horst Zuse
Dr.-Ing. 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, Faculty of Informatics, Postfach 4120, 39016 Magdeburg, Germany
Technical Editor: Dr. Achim Winkler
The journal is published in one volume per year consisting of two numbers.
The price 1997 for a single copy is DM 30,- for institutions and DM 20,- for private persons incl. postage handling.

EDITORIAL

This is the second issue of a new scientific journal in the field of software metrics and related quantitative aspects, the

METRICS NEWS.

The title was chosen to reflect the Journals attempt to summarize recent software metrics themes as position papers, chosen papers from our metrics workhops, and *news* (as information about the software metrics research area in the world, new books and conferences). The editors are working many years in the software metrics field and are specialized in measurement frameworks, function point analysis, measurement theoretical view, and practical applications.

The background of the METRICS NEWS contributors is the GI-interest group on software metrics founded in 1991. All members from the industry or academia are invited to present their experience or research results in the area of software quality assurance, software metrics, process management, software measurement frameworks etc.

The English language was chosen to reflect the international character of our research contacts and results embedded in European initiatives.

The editors are grateful to the Otto-von-Guericke University of Magdeburg for publishing this journal.

We hope that the new journal will be helpful to increase the awareness of the importance of software metrics issues in the improvement of software development processes and products.

The Editors

7. Workshop
SOFTWAREMETRIKEN

des Arbeitskreises Softwaremetriken und des Arbeitskreises Qualitätsverbesserung

Unser diesjähriger Workshop findet in der Zeit vom **18.09.** bis **19.09.1997** an der **Universität** in **Mannheim** statt. Bitte reichen Sie ganze Beiträge (maximal 10 Seiten) oder zwei- bis dreiseitige Abstracts zu folgenden Themen ein:

- Praktische Erfahrungen beim Einsatz von Softwaremetriken,
- Konzeptionen und Anwendungen von Erfahrungs-, Meß- und Projektmanagement-Datenbanken,
- Einsatz von Softwaremetriken für die Auswahl von Entwickungsmethoden (insbesondere objektorientierter Paradigmen),
- Neue Ansätze der Metrikenvalidation,
- Stand und Entwicklung von Metriken-Standards,
- Der objektorientierte Entwicklungsprozeß: Analyse, Bewertung und Verbesserung,
- Vorgehensweisen zur kontinuierlichen Qualitätsverbesserung,
- Vorhersagbarkeit objektorientierter Systeme (Schätzverfahren).

Der Workshop ist keinesfalls auf die genannten Themenstellungen beschränkt und soll darüberhinaus viel Spielraum für Diskussionen und Initiativen geben. Alle Beiträge sollen dabei als Vortrag oder als Auslage (Pinwand) zur Kenntnis gelangen. Die zum Vortrag ausgewählten Beiträge werden in einer geschlossenen Form publiziert.

Die Beiträge sind bis zum **18. Juli 1997** an eine der folgenden Adressen bzw. per Email zu senden:

- ***Kathrin Baumann***, (Leiterin der Arbeitskreises Qualitätsverbesserung), SAP-AG, Email: kathrin.baumann@sap-ag.de
- ***Prof. Franz Stetter***, (Workshop-Organisation), Universität Mannheim, A5, 68131 Mannheim, Email: fstetter@pi1.informatik.uni-mannheim.de
- ***Prof. Reiner Dumke***, (Leiter des Arbeitskreises Softwaremetriken), Universität Magdeburg, Fakultät für Informatik, IRB, Postfach 4120, 39016 Magdeburg, Email: dumke@irb.cs.uni-magdeburg.de

Quantitative Management of Software Process Improvement

Christof Ebert, Alcatel Telecom, Switching Systems Division, Antwerp

Quantitative data is crucial for understanding software development processes and to steer any reengineering activity. Quantitative management of a software process improvement (SPI) activity is not much different from managing a project. Unless supported by metrics, it is impossible to fully understand what is happening and what will be the outcomes of prospective changes. Quantitative management of SPI is thus concerned with identifying, measuring, accumulating, analyzing and interpreting project and process information for strategy formulation, planning and tracking activities, decision-making, and cost accounting.

Although the corporate metrics program has been set up and is maintained as part of the Division's SPI program, most benefits that we recorded are indeed related to project management:

- Improved tracking and control of each development project based on uniform mechanisms;
- Earlier identification of deviations from the given targets and plans;
- Accumulation of history data from all different types of projects that are reused for improving estimations and planning of further projects;
- Tracking process improvements and deviations from processes.

Metrics are obviously the key to successfully managing a SPI program because they link the improvement strategies, pilot results and various process reengineering efforts to the day-to-day business that after all keeps the company alive.

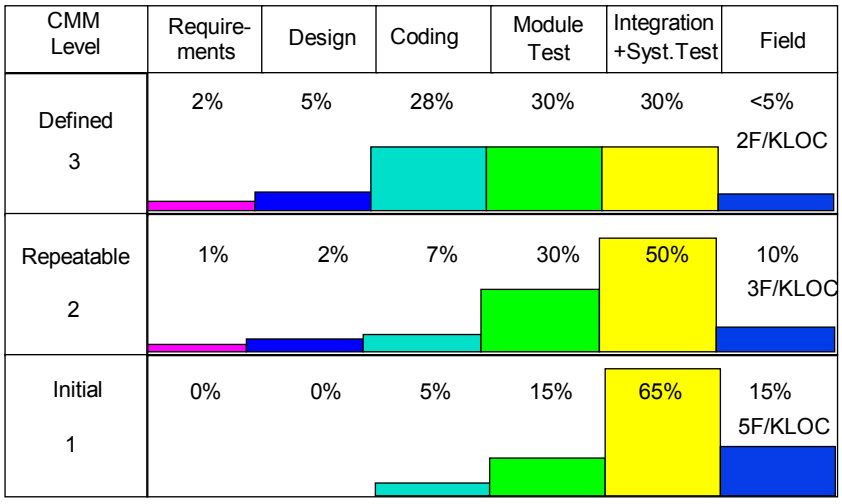
Appropriate Metrics for Different CMM Levels

CMM	Description	Metrics
5	Continuous improvements are institutionalized	Process metrics for the control of process change management
4	Products and processes are quantitatively managed	Process metrics for the control of single processes
3	Appropriate techniques are institutionalized	Defined and established product metrics; automatic metric collection
2	Project management is established	Defined and reproducible project metrics for planning and tracking (fault status, effort, size, progress); few process metrics for SPI progress tracking
1	Process is informal and ad hoc	Few project metrics (size, effort, faults); however metrics are inconsistent and not reproducible

Objectives related to individual processes must be unambiguous and agreed by the respective groups. This is obvious for test and design groups. While the first are reinforced for finding defects and thus focus on writing and executing effective test suites, design groups are targeting to delivering code that can be executed without defects. In case of defects they must be corrected efficiently, which allows for setting up another metric for a design group which is the backlog of faults it has to resolve.

It is thus important for process metrics to consider different viewpoints and their individual goals related to promotion, projects and the business. Most organizations have at least four: the practitioner, the project manager, the department head, and corporate executives. Their motivation and typical activities differ much and often create confusing goals which at the worst level are resolved on the practitioner level. Reuse for instance continuously creates trade-off discussions. When a project incurs expenses due to keeping components maintainable and to promote their reusability, who pays for it and where is it recorded in a history database that compares efficiency (e.g. bang per buck) of projects and thus of their management?

Typical benchmark effects of detecting faults earlier in the life cycle



The following key success factors could be identified while setting up a globally distributed metrics program:

- Start small and immediately. It is definitely not enough only to select goals and metrics. Tools and reporting must be in line; and all of this takes its time. It must however be clearly determined what needs to be measured before deciding based on what can be measured. Use external consultants where needed to get additional experience and authority.
- Motivate the metrics program with concrete and achievable improvement goals. Unless targets are achievable and clearly communicated to middle management and practitioners they will clearly feel metrics as yet another instrument of management control. Goals must be in line with each other and on various levels. Business goals must be broken down to project goals and those must be aligned with department goals and contents of quality plans. Clearly communicated priorities might help with individual decisions.
- Provide training both for practitioners who after all have to deliver the accurate raw data, and for management who will use the metrics. The cost and effort of training is often stopping its effective delivery. Any training takes time, money, and personnel to prepare, update, deliver, or receive it.
- Establish focal points for metrics in each project and department. Individual roles and responsibilities must be made clear to ensure a sustainable metrics program that endures initial SPI activities.
- Define and align the software processes to enable comparing metrics. While improving processes or setting up new processes, ensure that the related metrics are maintained at the same time. Once estimation moves from effort to size to functionality, clearly the related product metrics must follow.
- Collect objective and reproducible data. Ensure the chosen metrics are relevant for the selected goals (e.g. tracking because to reduce milestone delay) and acceptable for the target community (e.g. it's not wise to start with productivity metrics).
- Get support from management. Enduring buy-in of management can only be achieved if the responsibility for improvements and the span of necessary control are aligned with realistic targets. Since in many cases metrics beyond test tracking and faults are new instruments for parts of management this group must also be provided with the necessary training.
- Avoid abuse of metrics by any means. Metrics must be "politically correct" in a sense that they should not immediately target persons or satisfy needs for personal blames. Metrics might hurt but should not blame.
- The targets of any improvement program must be clearly communicated and perceived by all levels as realistic enough to fight for. Each single process change must be accompanied with the respective goals and supportive metrics that are aligned. Those affected need to feel that they have some role in setting targets. Where goals are not shared and the climate is dominated by threats and frustration, the metrics program is more likely to fail.
- Communicate success stories where metrics enabled better tracking or cost control. This

includes identifying metrics advocates that help in selling the measurement program. Champions must be identified at all levels of management, especially at senior level, that really use metrics and thus help to support the program. Metrics can even tie in an individual's work to the bigger picture if communicated adequately.

- Slowly enhance the metric program. This includes defining "success criteria" to be used to judge the results of the program. Since there is no perfect metrics program it is necessary to determine something like a "80% available" acceptance limit that allows to declare success when that is achieved.
- Don't overemphasize the numbers. It is much more relevant what they bring to light, such as emerging trends or patterns. After all the focus is on successful projects and efficiency improvement and not on metrics.

Time Table for Setting up a Corporate Metric Program

Activity	Elapsed time	Duration
Initial targets set up	0	2 weeks
Creation and kick-off of metric team	2 weeks	1 day
Goal determination for projects and processes	3 weeks	2 weeks
Identifying impact factors	4 weeks	2 weeks
Selection of initial suite of metrics	5 weeks	1 week
Report definition	6 weeks	1 week
Kick-off with management	6 weeks	2 hours
Initial tool selection and tuning	6 weeks	3 weeks
Selection of projects / metric plan	6 weeks	1 week
Kick-off with project teams / managers	7 weeks	2 hours
Collection of metric baselines	7 weeks	2 weeks
Metric reports, tool application	8 weeks	continuously
Review and tuning of reports	10 weeks	1 week
Monthly metric-based status reports within projects	12 weeks	continuously
Application of metrics for project tracking and process improvement	16 weeks	continuously
Control and feedback on metric program	24 weeks	quarterly
Enhancements of metric program	1 year	continuously

Metrics need to make sense to everybody within the organization who will be in contact with them. Therefore, the metrics should be piloted and evaluated after some time. Potential evaluation questions include:

- Are the selected metrics consistent with the original improvement targets? Do the metrics provide added value? Do they make sense from different angles and can that meaning be communicated without many slides? If metrics are considering what is measurable but don't support improvement tracking, they are perfect for hiding issues but should not be labeled metrics.

- Do the chosen metrics send the right message about what the organization considers relevant? Metrics should spotlight by default and without cumbersome investigations of what might be behind. Are the right things being spotlighted?

- Do the metrics clearly follow a perspective that allows comparisons? If metrics include

ambiguities or heterogeneous viewpoints they cannot be used as history data.

Software process improvement is now a big issue on the agenda of all organizations with software as a core business. As such it is also a major research topic, that may continue to grow in importance well into the 21st century. However, some software technologies have a shorter lifetime and for sure the management attention is focused rather on short-term achievements with impact to the score card. Unless tangible results can be achieved in the related short timeframe, interest in SPI will quickly wane.

Current Situation in Software Measurement Frameworks

Reiner R. Dumke
University of Magdeburg, Faculty of Informatics
Postfach 4120, D-39016 Magdeburg, Germany

1 Introduction

Measurement frameworks are in usually embedded in business perspectives such as [11]

- improving product delivery times,
- lowering software development costs,
- minimizing application backlog,
- improving skills level,
- assessing the value of consultants and contractors,
- optimizing the use of new technologies.

A main aspect for a successful application of a software measurement framework is the level of integration in the software process. Therefore, the existence of a software process model is an essential requirement for an efficient framework approach. Kinds of process models are ([5], [24]):

- *object management systems* (with the software process components: process, office, environment, resources, and interface; and the principles of co-operative entities on the basis of the service, object, and item level),
- *environment (tool) integrated facilities* (e. g. SPADE),
- *(design) process modelling languages* (such as PCTE-based, PML, and SOCCA),
- *formal approaches* (temporal logic-based, Petri nets, constraint-based, meta-process oriented, goal-oriented, etc.).

In this manner, we can establish the *informal* and *formal approaches* of software measurement frameworks.

2 Informal Approaches of Software Measurement Frameworks

The most applications of software measurement are goal-directed, informal approaches such as the goal question metric (GQM), the factor criteria metric (FCM), the quality function deployment (QFD), and the AMI (application of measurement in industry) approach [18]. Another kind of measurement frameworks are the process improvement models such as the CMM (Capability Maturity Model). Of course, the GQM can also be used for process improvement, but the GQM is a general approach (also for product and resources evaluation or for special development aspects). The informal approaches of software measurement frameworks consist of the following general components

- textual descriptions/questions,
- rules, "laws" and experience notices,
- standards.

Textual descriptions including some general remarks on software measurement are ([14], [23])

- the *ISO 9000-3 standard,*
- the *Software Quality Metrics* report (FAA Technical Center, New Jersey),
- the *TickIT approach* (UK),
- the *BOOTSTRAP quality standard* (ESI, Esprit project),
- the *Software Measurement Guidebook* (NASA),
- the *Trillium standard* (Bell Canada),
- the *AQAP* and the *DOD STD 2167A* (USA military area),
- the European *SPICE project.*

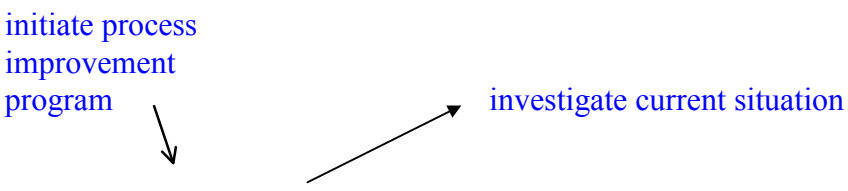
An example for the underlying rules in this software (quality) measurement is given in the NASA Guidebook [21]:

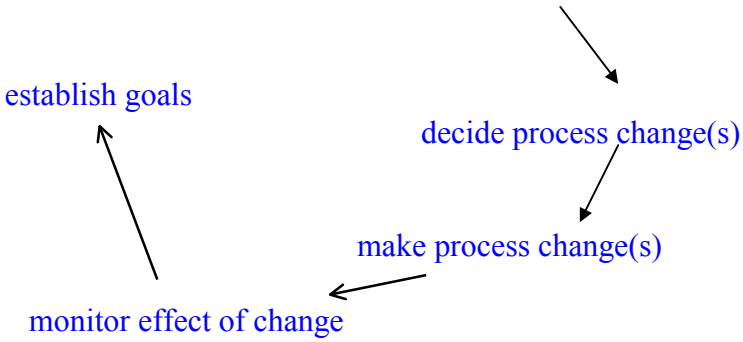
- establishing a ***measurement program*** (including the definition of the goals, the responsibilities and selecting the measures),
- core measures (especially the costs, errors, process characteristics, project dynamics, and project characteristics),
- operation of the measurement program (use of metrics tools, storing the measurement values etc.),
- analysis, application, and feedback (as goal of the software process or product improvement).

The software measurement itself can be divided in the main components [18]

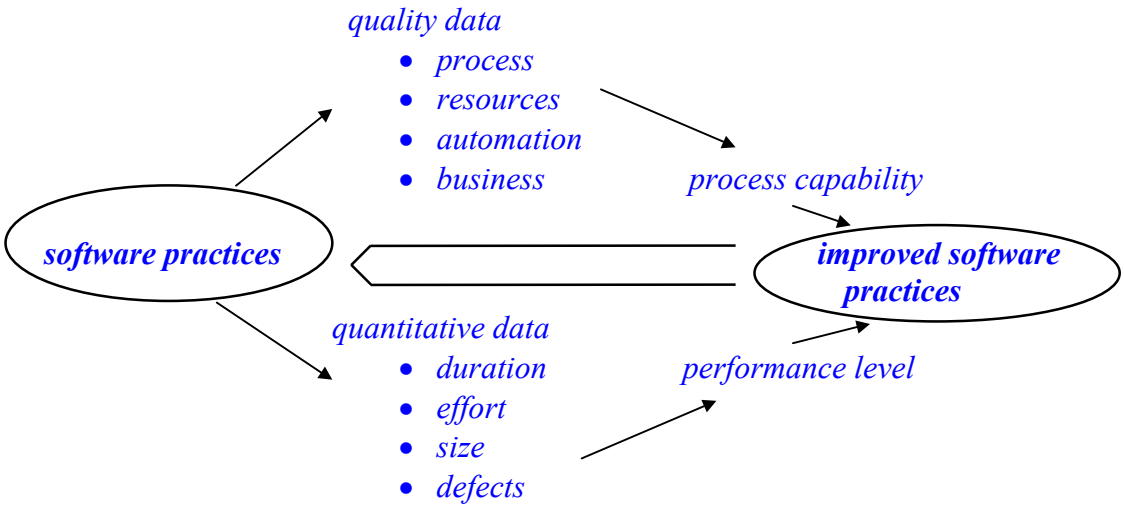
> *entities, attributes, relationships between entities and attributes, units, scale types, values, properties of values, the origin of values, defining measures.*

Another approach of the software measurement for the process improvement is described by Kitchenham [18] in the cycle





In the same manner, Garmus and Herron [11] define the *complete process assessment model* as



Here, we can see the very general characteristics of these approaches: Reasoning in the wide and apparently diverse range of topics that cover the software measurement, such as [4] cost estimation models, productivity models, quality control and assurance, data collection, quality models and measures, reliability models, performance measurement, and structural and complexity metrics.

The probably best-known measurement framework is described in the paper of Basili et al [1] as general aspects of the ***experimentation in software engineering*** in the definition of the experiment (motivation, object, purpose, perspective, domain, and scope), the planning of the experiment (design, criteria, and measurement), the operation of the experiment (preparation, execution, and analysis), and the interpretation of the experiment (as interpretation context, extrapolation, and impact). This framework of software experimentation is a good checklist for controlling the completeness of an experiment (see also [3]), but it allows for more than hundred variants of experiments.

In [20] McGregor defines an *"Iterative Incremental Metric Model"* that requires a refinement in the application of software metrics. The main thesis in this approach are that

- a metric can be specified, in terms of what attribute it represents, independent of any specific implementation of the metric,

- various definitions can be sequenced to provide continuous measurements of an attribute across the phases in the lifecycle,

- there is an acceptable trade-off between the precision of the calculation of the value and the availability of an estimate of the value earlier in the lifecycle.

The approach of McGregor also retains the relationship of the measurement framework with the software process model.

3 Formal Approaches in Software Measurement

Formal approaches for software measurement frameworks can be divided in *algebraic approaches, axiomatic approaches, functional approaches,* and *rule-based approaches*. In the following we explain some examples of these approaches.

Algebraic approaches of measurement: one example is given by Shepperd in [23] and includes the general formal description as

- an algebraic description of the measured model (*mod* stands for module)
  $new: \rightarrow design$
  $add: mod \times design \rightarrow design$

- a general description of a metric
  $metric: design \rightarrow nat$

- a special description of a concrete metric (e. g. module counting)
  $m: mod$
  $D: design$
  $metric(new) = 0$
  $metric(add(m,D)) = 1 + metric(D)$

The article gives a full description of a module-based system design metric including the fan-in and fan-out characteristics.

Axiomatic approaches of measurement: a (classical) axiomatic approach is given by Prather in [22]. The basic elements are the restricted program constructs of the structured programming (the sequence, the selection, and the repetition). On this basis a (complexity) measure was defined as

- $measure(sequence) = term_1$,

- $measure(selection) = term_2$,

- $measure(repetition) = term_3$.

The description of the measures includes the value for a simple statement. The value of a program is derived by the use of the three axioms above.

Another axiomatic approrch is given by Zuse in [25] (see also [26]) based on measurement theory. The main idea is the definition of an *empirical relational system* and a *numerical relational system.* Software measurement is described as the homomorphism

$$object_1 \geq_{empirical} object_2 \Leftrightarrow measure(object_1) \geq_{numerical} measure(object_2)$$

The axioms of the weak order, the (weak) associativity, the (weak) commutativity, the (weak) monotonicity, and the Archimedean axiom help to determine the scale types of a concrete software measure. This approach supports the full characterization of a measure including the correct application of statistical analysis methods.

The axiomatic approach of Fenton in [10] includes

- a *prime-based definition* of program components,
- the metric execution for the *sequencing* of primes,
- the metric execution for the *nesting* of primes.

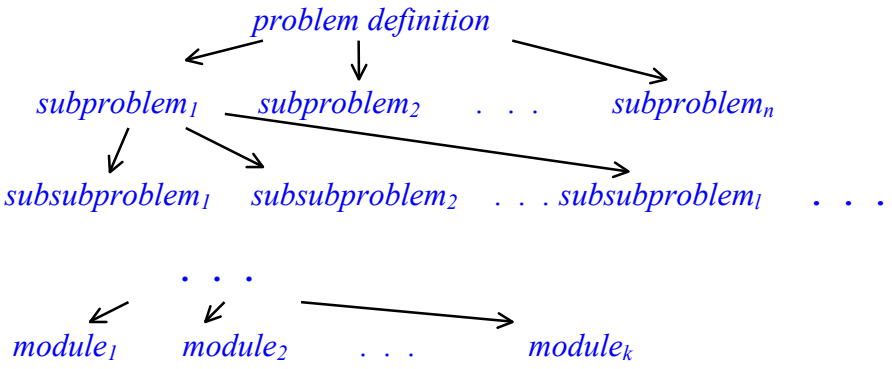
The software measurement itself is directed at the *process, product,* and *resources.* These components are characterized by *internal* and *external attributes,* and the measurement is divided in *assessment* and *prediction.* The areas of software measurement are the cost and effort estimation, the productivity measures, the quality control and assurance, the data collection, the quality models and measures, the reliability models, the performance evaluation, the algorithmic/computational complexity, and the structural and complexity measures.

Functional approaches of measurement: an example of a functional measurement approach is COCOMO of Boehm (see also [2]) with the main formula as

$$effort = \alpha\, LOC^{\beta}$$

α and β have special values for special project characteristics. The problem is to estimate the lines of code (LOC). The formula is a summarizing of experience of the software development effort.

Another functional approach of software measurement is given by Ejiogu in [8] and is based on the problem refinement in the following manner



The functional approach consists in the definition of the measures as formulas such as *height of the tree,* and characterize the monadicity, the entropy, the cohesion and coupling of the modules, the degree of refinement, the modularity, the maintainability, the test coverage, the reliability, and the level depended productivity.

A further functional approach of measurement is the function point method of Albrecht (see also in [16]) that was based on the (weighted) assessment of

outputs, inquires, inputs, files, and *interfaces*

for every (software) component and the final calculations with an adjustment factor to the final function points. In further experiments function points have been mapped to the effort of the software product development.

Rule-based approaches of measurement: one example of this approach is given by Hausen in [13] and is based of the definition of rules for the given (software) product in the form

IF $predicate_1$ $predicate_2$ . . . $predicate_l$ *activity* $estimation_{expert}$ *predicate*

THEN $volume_{component}$ *predicate*

where the rules are defined for the *component* as *activities, objects, functions, data, procedures,* and *variables.* In the same manner quality rules related to the special software components have been established.

Another "rule-based" approach consists of formal language rules by Jacob and Cahill in [15] as attribute grammar approach. The metrics rules are defined in the attributes and the underlying semantic functions. Such attributes are for example

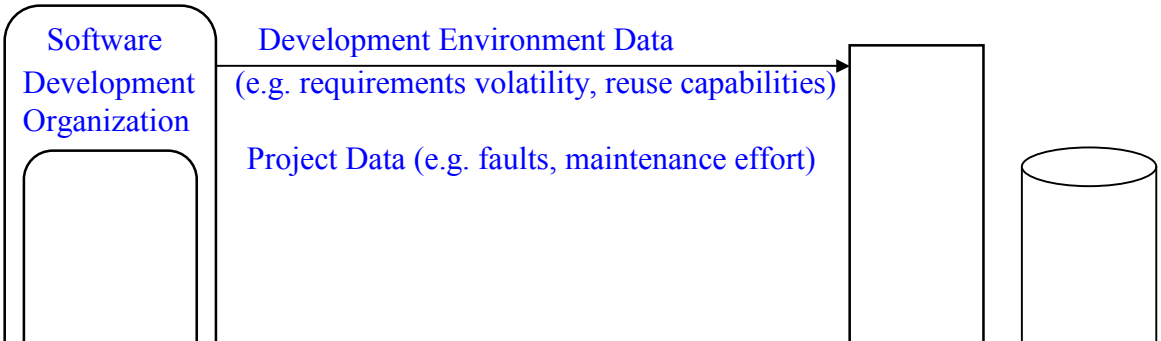
statement counting: $p \rightarrow statement_{increment\ scount}$

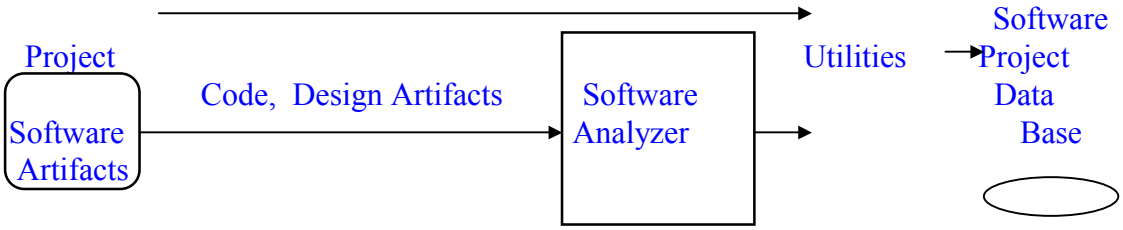
decision counting: $p \rightarrow if\text{-}statement_{increment\ dcount}$
$p \rightarrow while\text{-}statement_{increment\ dcount}$
etc.

The semantic function includes the final execution of the defined metric (for example as a multi-dimensional form etc.).

4 Statistical Analysis of Measurement Data

A statistical analysis approach can be characterized by the following general scheme of Evanco and Lacovara for the data collection and analysis [34]:





The applicable statistical methods are given in the following table (see also [12]).

Type of methodology	Application
Ordinary least squares regression models	Subsystem defects or defect densities
Poisson models	Library unit aggregation defect analysis
Binomial analysis	Defect injection probabilities
Ordered response models	Defect proneness
Proportional hazards models	Failure Analysis incorporating software characteristics

In order to use statistical methods it is necessary to know the scale type of the measurement data. For the correlation methods we must guarantee the following relations

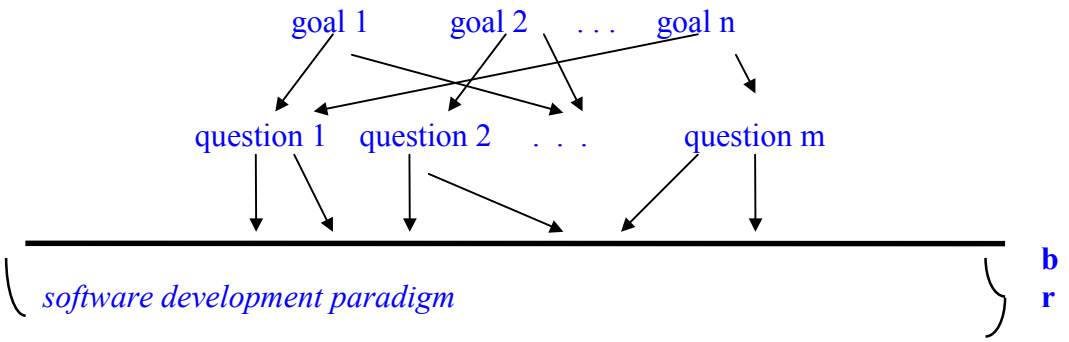
scale type	correlation coefficient
ordinal scale type	Kendall or Spearman correlation
interval scale type	Pearson or multiple correlation
ratio scale type	Pearson, multiple, and variance

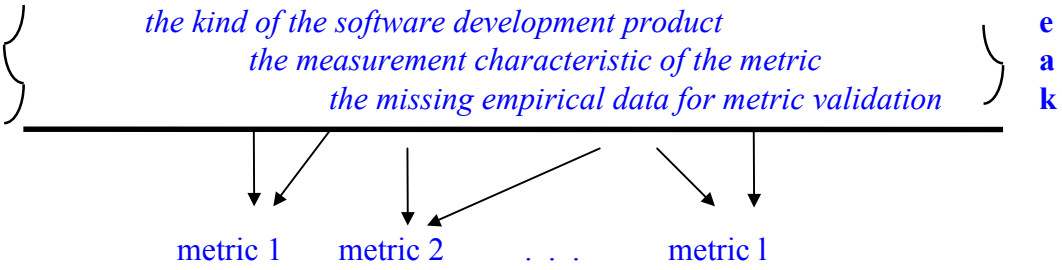
Other approaches include the use of classification methods such as *pareto classification, factor-based discriminant analysis, fuzzy classification* or *neural network approaches* ([7], [17]).

Note, that an essential aspect of the (statistical) measure analysis is given by the metrics validation as a *statistical* or *application validation* and as a *predictability validation* [23].

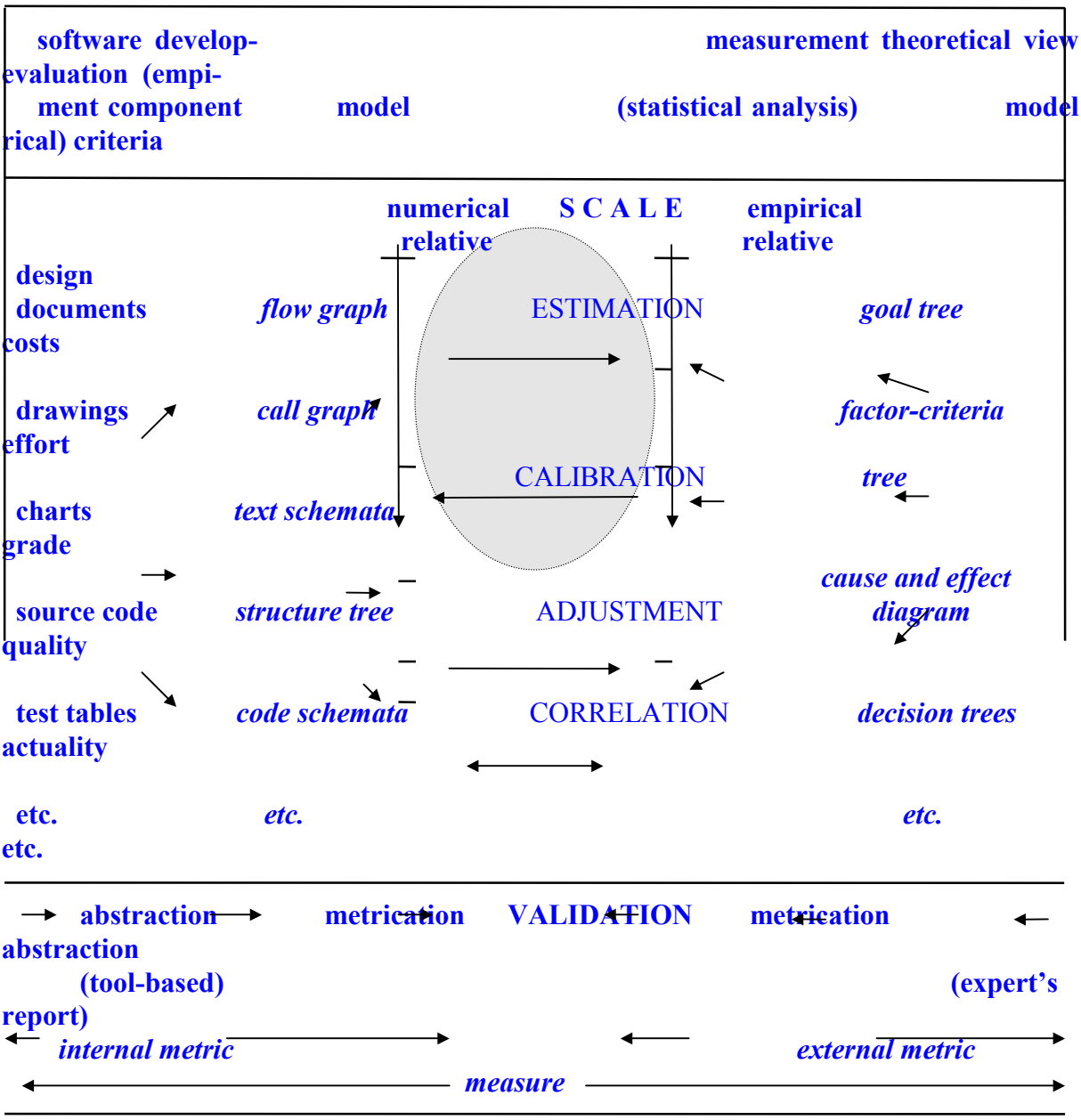
5 Benefits and Weaknesses of Informal and Formal Measurement Approaches

The problem of the use of informal approaches in general is the ***break*** between the (subdivided) quality aspects and their quantification, e. g. in the GQM





This situation is also given in the AMI, CMM, ISO 9000 and the other informal approaches. A general description of this "break" can also be seen in the following validation schema of software measurement application in the grey area [6].



The weakness of the evaluation part in the measurement frameworks results in different kinds of empirical evaluations as:

1. (vague) expertise's,
2. defining limits (warning or dangerous values),
3. (ordinal) empirical evaluations,
4. (full) empirical function (e.g. function points etc.).

In most applications only the first and the second kind of the empirical evaluation are used.

A summarized overview of advantages and shortfalls of formal approaches is given in the following table.

approach	benefits	weakness
algebraic	a well-defined metrics algebra	no independence of the development paradigm
axiomatic	an exact definition of the metrics characteristics	only a few practicable results
functional	compact definition of experience	problem in their use for new development paradigms
rule-based	a well-defined metrics language	only a few empirical evaluations

Some measurement and evaluation frameworks problems in general are (see also [19] and [26]): *no experience for modern or new paradigms, no support for adjustments, no characteristics of the given or used (acquisited) software, no exact knowledge about the applied metrics.*

References

[1] Basili, V.R.; Selby, R.W.; Hutchens, D.H.: *Experimentation in Software Engineering.* IEEE Transactions on Software Engineering, 12(1986)7, pp. 733-743

[2] Boehm , B.W.: *Software Risk Management*. IEEE Computer Society Press, 1989

[3] Bourque, P.; Maya, M.; Abran, A.: *A Sizing Measure for Adaptive Maintenance Work Products.* Proc. of the IFPUG Spring Conference, Atlanta, April 22-26, 1996

[4] Bush, M.E.; Fenton, N. E.: *Software Measurement: A Conceptual Framework.* Journal of the Systems and Software, 12(1990), pp. 223-231

[5] Daskalantonakis, M.K.: *A Practical View of Software Measurement and Implementation Experiences Within Motorola.* IEEE Transactions on Software Engineering, 18(1992)11, pp. 998-1010

[6] Dumke, R.; Foltin, E.; Koeppe, R.; Winkler, A.: *Measurement-Based Object-Oriented Software Development of the Software Project "Software Measurement Laboratory".* Preprint 1996, University of Magdeburg (40 p.)

[7] Ebert, C.: *Evaluation and Application of Complexity-Based Criticality Models.* Proc. of the Third International Software Metrics Symposium, March 25-26, Berlin, 1996, pp. 174-185

[8] Ejiogu, L.O.: *Software Engineering with Formal Metrics.* QED Technical Publ., 1991

[9] Evanco, W:M:; Lacovara, R.: *A Model-Based Framework for the Integration of Software Metrics.* The Journal of Systems and Software, 26(1994), pp. 77-86

[10] Fenton , N.: *Software Metrics - a rigorous approach.* Chapman & Hall, 1991

[11] Garmus, D., Herron, D.: *Measuring the Software Process - a practical guide to functional measurements.* Prentice-Hall Publ., 1996

[12] Han, K.J.; Yoon, J.; Kim, J.; Lee, K.: *Quality Assessment Criteria in C++ Classes.* Microelectronics Reliability Journal, 34(1994)2, pp. 361-368

[13] Hausen, H.: *A Rule-Based Approach to Software Quality Engineering.* in: Fenton/Littlewood: Software Reliability and Metrics, Elsevier Publ., 1991, pp. 48-68
[14] Henderson-Seller, B.: *OO Metrics Programme.* Object Magazine, October 1995, pp. 73-95
[15] Jacob, P.; Cahill, T.: *Software Product Metrics as Attributes in an Attribute Grammar.* Proc. of the 2ICSQ, October 1992, Research Triangle Park, USA, pp. 40-49
[16] Jones ,C.: *Applied Software Measurement.* McGraw-Hill, 1991
[17] Khoshgoftaar, T.M.; Szabo, R.M.: *ARIMA models of software system quality.* Proc. of the Annual Oregon Workshop on Software Metrics, April 10-12, 1994, Oregon
[18] Kitchenham, B.: *Software Metrics - Measurement for Software Process Improvement.* NCC Blackwell Publ., 1996
[19] Kitchenham, B.; Pfleeger, S.L.; Fenton, N.E.: *Towards a Framework for Software Measurement Validation.* IEEE Transactions on Software Engineering, 21(1995)12, pp. 929-944
[20] McGregor, J.D.: *Managing metrics in an iterative environment.* Object Magazine, October 1995, pp. 65-71
[21] NASA : *Software Measurement Guidebook.* Maryland, 1995
[22] Prather, R.E.: *An Axiomatic Theory of Software Complexity Measure.* The Computer Journal, 27(1984)4, pp. 340-347
[23] Shepperd, M.: *Foundations of Software Measurement.* Prentice Hall Publ., 1995
[24] Warboys, B.C. (Ed.): *Software Process Technology.* Proc. of the EWSPT'94, Springer Publ., Lecture Notes on Computer Sience 772, 1994
[25] Zuse , H.: *Software Complexity - Measures and Methods.* de Gruyter Publ., 1991
[26] Zuse, H.: *A Framework of Software Measurement.* to be published

An email information

Fernando Brito e Abreu, INESC - MOOD Project Leader, Lisbon, Portugal

We are actively working on MOODKIT G2 (second generation) which is radically different from previous on (G1). Among the improvement is the ability of metrics capture either by forward (from models in a CASE TOOL) or reverse engineering (from source code in several OO languages). MOODKIT G2 relies on an intermediate OO design language named GOODLY (a Generic Object Oriented Design Language? Yes!).

The GOODLY language is up and running! A GOODLY specifications hypertext browser with high traceability capabilities and several source code examples that were generated with MOODKIT G2 (under construction) are now available at our web site. This bowser will soon show the calculated MOOD metrics values. The MOOD set is being currently reviewed and expanded.

The MOOD Project WWW server is located at the following address:

http://albertina.inesc.pt/ftp/pub/esw/mood

Please use a browser that supports frames (e.g. Netscape 2.0 or later releases).

PRODUCT	STATUS	AVAILABILITY
GOODLY specifications parser and linker	Ready	available on request
GOODLY specifications browser	Ready	use it in the web
GOODLY to Smalltalk converter	2 nd week May	(forecast)
Smalltalk to GOODLY converter	2 nd week May	(forecast)
Eiffel to GOODLY converter	3 rd week May	(forecast)
OMT (ParadigmPlus) to GOODLY converter	3 rd week May	(forecast)
MOOD metrics extraction from GOODLY code	4 th week May	(forecast)
Java to GOODLY converter	4 th week May	(forecast)
C++ to GOODLY parser	2 nd week June	(forecast)
Object Pascal (Delphi) to GOODLY parser	4 th week June	(forecast)

The MOOD team is waiting for your feedback and your cooperation plus!

The MOOD (Metrics for Object Oriented Design) metrics originated from the PhD research work carried out by Fernando Brito e Abreu, enriched by contributions of many others, either originated within the MOOD team or organization where MOOD project team is hosted, see our central web site (http://www.inesc.pt).

The MOOD project is an academic project, not a commercial one! The only thing we ask from you is to share with us the results you got with our tools and your constructive contributions on improving and/or extending the MOOD metrics set. In particular we seek cooperation with reals industrial projects where process data (schedules, effort, defect reports, etc.) are available, in order to construct empirical validation studies, as well as academic theoretical validations ones.

ISBSG - A worldwide Software Measurement Initiative

The ISBSG (International Software Benchmarking Standards Group) had its origins in the work performed by the Australian Software Metrics Association (ASMA) in software benchmarking. In 1990, a Special Interest Group in ASMA met to develop a practical industry standard for quantifying the output from software projects. This led to the establishment of a repository of data on Australian projects in 1992.

The success of this initiative created considerable international interest. In June 1994, the software metrics organisations of New Zealand (SMANZ), the United Kingdom (UFPUG), and the United States (IFPUG), together with ASMA, formed ISBSG. Later other metrics organisations (for instance from Canada, Germany, France) became involved. The ASMA model was used for a de facto international standard. Through ISBSG, the various associations and their members can collect and share data to facilitate international benchmarking. The actual fourth release of the Benchmarking Repository contains data collected from 396 projects from 14 countries.

The ISBSG Repository is based on the following principles:

- **Practitioner Driven** and **Practitioner Accessible**: Each IT-organization, whether they are members of their respective national metrics organisation or not, may contribute to the ISBSG Repository and use the services of ISBSG.
- **Independence** from vested business and research interests whenever they are liable to compromise the objectives of the Repository.
- **Integrity** of the Repository data must be maintained through the application of rigorous procedures.
- **Confidentiality** of the contributors.

The establishment of the ISBSG Repository has made it possible to offer the industry a number of services:

- *The Repository* itself can be used as an alternative to In-house metrics databases
- *A Project Benchmarking Profile Report* is sent back to the contributor. It compares the submitted project with others of the same class within the repository
- *Best Practice Networking* is available for contributors
- *Organisational Benchmarking* is available to organisations to compare themselves against similar organisations
- *ISBSG Releases* (reports on the ISBSG Repository)
- *Customised Analysis and Reports*

ISBSG is working permanently to increase the value of the services offered. At around nine month intervals interested members meet at the ISBSG workshop. At the last workshop, held in conjunction with the IFPUG'97 Spring Conference, two research contracts with the Monash University (Australia) and the Université du Québec à Montréal (Canada) have been initiated.

If you want to learn more about the ISBSG initiative or how to contribute to the ISBSG Repository please see **http://www.bs.monash.edu.au/asmavic/isbsg.htm**.

SMLab's WorldWideWeb Project

The Software Measurement Laboratory of the University of Magdeburg was established to support the Software Metrics efforts of the (local) IT community and to conduct university research and education. As a service for the public, SMLab maintains a Website to inform about new devlopments and to provide a world-wide discussion platform.

In the position paper *Current Situation in Software Measurement Frameworks* beginning on Page 11 of this issue, the author mentions a break between the quality aspects and their quantification with metrics. For the Software Metrics field, a science that is largely dominated by empirical results, conducting experiments and analysing the results is a critical and important step toward the formation of valid models.

In order to provide an overview about experimental results the Software Measurement Laboratory has added a summary of software measurement experiments to its Web-site. The more than fifty eperiment descriptions are grouped in

- *Software Process Experiments* (Process Maturity, Process Management, and Process Life Cycle Experiments)
- *Software Product Experiments* (Size, Architecture, Structure, Quality, and Complexity Experiments)
- *Software Resource Experiments* (Personnel, Software, and Hardware Experiments)

"Classical" Experiments as Halsteads Experiments to the definition of his "Software Science" are included as well as more recent experiments on Object Oriented Programming or World Wide Web design. For every experiment, a reference for further reading is provided. The Software Measurement Laboratory invites you to contribute your experience and experiment to make your results accessible to the software engineering community.

Another point of interest for the practitioner in the software metrics field is the application of Computer Assisted Measurement and Evaluation (CAME) Tools. Based on a general software measurement framework the Web Site contains a short description and evaluation of the better know measurement tools used in the European market.

Some sample on-line applications are available to demonstrate the capabilities offered by hypermedia technologies.

The Web-Site of the Software Measurement Laboratory can be found at:

http://irb.cs.uni-magdeburg.de/se/

Abran, A.; Dumke, R.; Lehner, F.: Software Metrics

Gabler-Verlag, Wiesbaden, 1997

This book contains all presentations of the 1996 workshop of the GI-interest group on software metrics. It will be available by the end of June 1997.

Kitchenham, B.: Software Metrics-Measurement for Software Process Improvement

NCC Blackwell Publ., 1996

This book explains how software measurement can be used to support software process improvement by providing objective methods of characterizing process capability and evaluating the effect of process changes.

This Book sets out an approach to the formel validation of measures and the theory of statistical data analysis for students. It also contains, for practitioners, many examples of the use of real data in real projects.

Poulin, J.S.: Measuring Software Reuse

Addison-Wesley, 1997 (195 p.)

With the techniques in this book, you will have the tools you need to design a far more effective reuse program, prove its bottom-line profitability, and promote software reuse within your organization. Measuring Software Reuse brings together all of the latest concepts, tools, and methods for software reuse metrics, presenting concrete quantitative techniques for accurately measuring the level of reuse in a software project and objectively evaluating its financial benefits.

* **Third Australian Conference on Software Metrics 1996 (ACOSM 96)**

  The third Australien Conference on Software Metrics 1996 (ACOSM 96) took place in Melbourne / Australia from Tuesday November 19 to Thursday, November 21, 1997. About fifty people did join the conference in the very impressive Hotel Sofitel in downtown Melbourne. The conference was organized by the Australian Computer Society (ACS).
  The objectives of the conference were Breaking Performance Barriers.
  The first day was a tutorial day. P. Goodman gave a tutorial about the implementation of a software metric program, and C. Symons presented a comparison of the traditional Function Point Method and the MARK II Method.
  On Wednesday, C. Symons gave a keynote presentation of the future of size measurement and Brian Henderson-Sellers followed with a keynote presentation of research ideas and practical experiences of object-oriented measurement. On Thursday, Horst Zuse presented fundamental concepts of measurement in a keynote presentation.
  Other speakers came from New Zealand, Norway, UK, and - of course - Australia.
  My personal impression is that the companies in Australia, but also the universities, are very active in improving software quality by quantitative methods of software engineering.

* **First Euromicro Working Conference on Software Maintenance and Reengineering,**

  March 17-19, 1997, Berlin

* **Fourth International Software Metrics Symposium**

  March 1997, Boston (incl. with the ICSE'97)

* **3rd International Conference on Reliability, Quality & Safety of Software-Intensive Systems (ENCRESS'97)**

  May 29-30, 1997, Athens, Greece

* **Fifth International Symposium on Assessment of Software Tools and Technologies**

  June 3-5 1997, Pittsburgh

* **European Software Control and Metrics Conference**

  was continued after the Wilmslow (May 1996) Conference

* **Seventh International Conference on Software Quality in New Orleans (ICSQ'97)**

  October 1997, New Orleans

* *metrics themes are also discussed in the yearly* **OOIS**, **ECOOP** *and* **ESEC** *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

- **http://www.sei.cmu.edu**
  SEI Pittsburgh

- **http://dxsting.cern.ch/sting/sting.html**
  STING: News Browser, Glossary Search, Projects and Measurement Tools at CERN

- **gopher://gopher.cs.tut.fi/11/pub/src/software-eng/metrics**
  C Metrics Package

- **http://www.spr.com/**
  Software Productivity Research, Capers Jones

- **http://fdd.gsfc.nasa.gov/seltext.html**
  SEL-Homepage

- **http://www.qucis.queensu.ca/Software-Engineering/Cmetrics.html**
  Queens University of Canada

- **http://www.esi.es**
  ESI Spain

- **http://saturne.info.uqam.ca/labo_Recherche/lrgl.html**
  University of Quebec

- **http://www.SoftwareMetrics.com**
  IFPUG Information by David Longstreet

- **http://www.utexas.edu/coe/sqi/**
  Software Quality Institute, University of Texas at Austin

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

- **http://www.inesc.pt/index-eng.html**
  Metrics for Object Oriented Design (MOOD) Project Team and the ftp://albertina.inesc.pt/pub/esw/modd
  MOOD-Server

- **http://divcom.otago.ac.nz:800/com/infosci/smrl/home.htm**

- **http://www.irb.cs.uni-magdeburg.de/se/**
  Software Meßlabor der Universität Magdeburg

- **http://www.cs.tu-berlin.de/~zuse**
  Arbeitsgruppe Softwaremetriken

- **http://www.sbu.ac.uk/~csse/publications/OOMetrics.html**
  Object-Oriented Metrics

- **http://www.sbu.ac.uk/~csse/ami.html**
  ami - Application of Metrics in Industry

- **http://www.dfn.de/~atw/bmbf/foerderprogramme/swt/SWT.html**
  Initiative zur Förderung der Software-Technologie in Wirtschaft, Wissenschaft und Technik

- **http://www.iso.ch/9000e/forum.html**
  The ISO 9000 Forum

- **http://ceswww.utexas.edu/sqi**
  Software Quality Institute (SQI)

- **http://www.tiac.net/user/pustaver/**
  The Software Quality Page

- **http://www.theriver.com/qa-inc/**
  Quality America, Inc's Home Page

- **http://www.ele.vtt.fi/docs/aslehti/magaz_z.htm**
  A primer for total quality in software development

- **http://www.nist.gov/quality_program/**
  NIST Quality Program

- **http://www.quality.org/qc/**
  Quality Resources Online

- **http://www.almaden.ibm.com/journal/sj33-1.html**
  IBM Systems Journal - Software Quality

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

News Groups

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

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

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

METRICS NEWS

VOLUME 2 **1997** **NUMBER 1**

CONTENTS

ISSN 1431-8008