Evaluated Object-Oriented Software Development

Reiner R. Dumke and Erik Foltin

University of Magdeburg,
Postfach 4120, D-39016 Magdeburg,
Germany

E-mail: {dumke,foltin}@irb.cs.uni-magdeburg.de
Fax: +49-391-67-12810

ABSTRACT: *This paper describes the fundamental ideas of our present project - the Software Measurement Laboratory - as a method of metrication of the object-oriented software development. The underlying measurement framework starts at the first step of the software development (the problem definition) and measures the metric mutations in the object-oriented paradigm of Coad/Yourdon and the implementation in Smalltalk and C++. The object-oriented software development was described with development indicators.*

KEY WORDS: *software measurement, object-oriented software development, software quality assurance*

1. Introduction

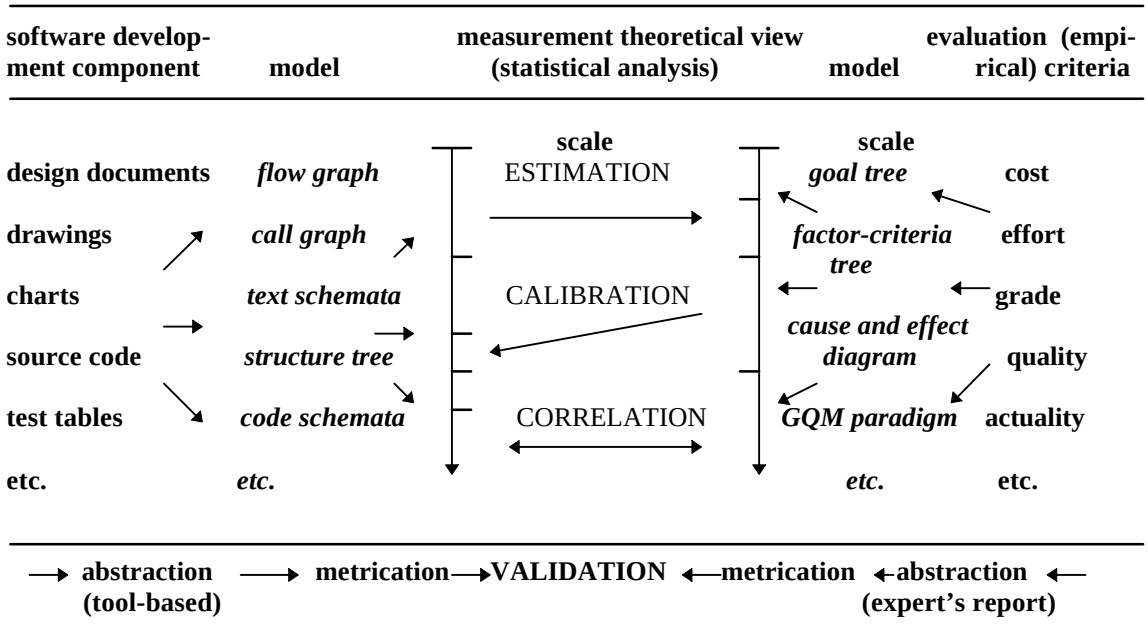
The use of a new software development paradigm in general starts with a lot of persuasive (but not proved) hypotheses. Therefore, from the software engineering point of view the goal of the measured object-oriented software development (OOSE) can be characterized as:

1. the understanding of the special development method or paradigm,
2. the measurement based method comparison, and the controlled software development process in the manner of the Capability Maturity Model of the Carnegie Mellon University in Pittsburgh.

The **recent works** in software measurement for object-oriented software development can be shortly characterized as

- **statistical analysis** by Rocache /Rocache 89/ of elements of an object-oriented development system (Smalltalk-80), or by Szabo and Khoshgoftaar of a C++ communication system (/Szabo et al 95/),
- **metrics set definitions** by Abreu /Abreu et al 94/ for C++ with the two vectors the category (design, size, complexity, reuse, productivity, and quality), the granularity (system, class, and method), by Binder /Binder 94/ in a set of C++ metrics to measure the encapsulation, the inheritance, the polymorphism and the complexity, by Arora et al for the real-time software design in C++ (/Arora et al 95/), and by Lorenz as a metrics set that can be used for both languages (C++ and Smalltalk, /Lorenz 93/),
- **metrics for aspect measurement** by Ott et al of the class cohesion (/Ott et al. 95/), or by Bieman or John of the reuseability (/Karunanithi et al 93/, /John et al 95/), and by Lejter of the maintenance (/Lejter et al 92/),
- **information theoretical approaches** in the measure of the conceptual entropy by Dvorak /Dvorak 94/ or in the cognitive approach by Henderson-Sellers et al /Cant et al 94/ with the landscape idea along the method routes or the learnability aspects in the use of class libraries (/Lee et al 94/), and

- **validation of enclosed approaches** by Chidamber and Kemerer /Chidamber et al



94/ as an apporach of a metrics definition based on a measurement theoretical view (with a viewpoint as empirical attribute), the extensions of these measures by Li et al (/Li et al 95/), the analysis of Churcher and Shepperd (/Churcher et al 95/), and the investigations of Zuse (/Zuse 94/).

These and the other concepts are first steps in a global measurement approach for the object-oriented software development and are missing (see also /Jones 94/) the evaluation of the continuity of the object-oriented software development process: the object-oriented analysis (OOA) ⇒ object-oriented design (OOD) ⇒ object-oriented programming (OOP) and the possibility of the reverse process (!).

2. The OOSE Measurement Framework

2.1. The general approach

The principal ideas of this measurement framework are given in /Dumke et al 94/ and are related to the object-orientation to understand and to quantify the chosen method. A standardized metric set for OOSE does not exist (only a metrics definition standard /IEEE 93/). Therefore, it is necessary to define metrics and to analyse them. The validation problem is the main problem in the application of software metrics (see the figure above). The software measurement is directed to the three main components in the (object-oriented) software development (see also /Fenton 91/)

- the ***process measurement*** for understanding, evaluation and improvement the development method,
- the ***product measurement*** for the quantification of the product (quality) characteristics and validation of these measures,
- the ***resource measurement*** for evaluate the support (CASE tools, measurement tools etc.) and the chosen implementation system.

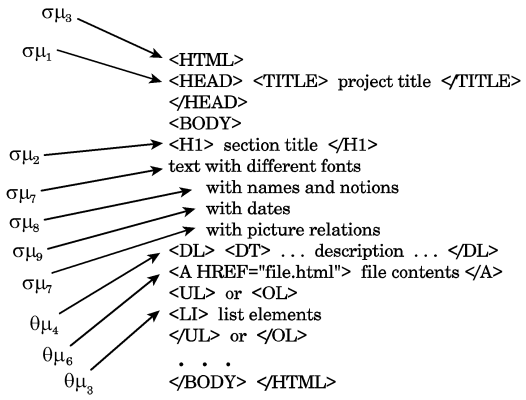
Some main ideas and some short results of an application on the Software Measurement Laboratory of the University of Magdeburg (SMLAB) are given in the following (see also in the WWW in

http://irb.cs.uni-magdeburg.de/ se/metrics_eng.html).

2.2. The Process Measurement

The chosen OOSE method is the Coad/Yourdon approach (described in /Coad et al 93/) and begins with the transformation of the problem definition in a graphical representation with an underlying documentation. The documentation stores all information that cannot be presented in the drawing. The drawings (also possible in some variants) and the documentation constitute the OOA model. In a first evaluation of this method we can establish as goals of the process measurement and the realized activities:

How can we measure the object definition process? This question leads us to the first step of the software development - the problem statement. We need a computational stored problem definition to measure the object definition. The SMLAB problem definition must be presented for all members of the software engineering team



and the document itself is an essential source for many outputs such as milestones in the different investigations, overview for some administrations. Therefore, we decided to use a local net html file set of the World-Wide Web as a ***living document system***. The elements of our problem statement are a

- ***list of contents*** as
  - problem description,
  - constraints,
  - given situation,
  - functional requirements,
  - management requirements (controlling and quality) and a
- ***list of components*** as
  - notions, names,
  - dates,
  - pictures, and
  - (hypertext) relations.

The process measures for the problem definition are (in a first step)

- the number of notions, names or titles,
- the number of dates (times and events).

The measure mutation was analysed, for example in the problem definition (#notions/names) to the number of the class definitions in the model and in the implementation. Further measurements are related to the adjectives/predicates into the class attributes or variables, verbs/adverbs into the class services or methods and dates/constraints into the model documentation and implementation.

We can establish the relation of 4600 notions (names or titles) into 76 object classes. Note, that a lot of notions in the problem definition are instances of the defined classes. So, we get the ***specification indicators as***

class definition indicator (CDI) as

$$\frac{\textit{number of notions}}{\textit{number of defined classes}}$$

attribute definition indicator (ADI) *as*

$$\frac{\textit{number of adjectives or predicates}}{\textit{number of defined attributes}}$$

service definition indicator (SDI) *as*

$$\frac{\text{number of verbs or adverbs}}{\text{number of defined services}}$$

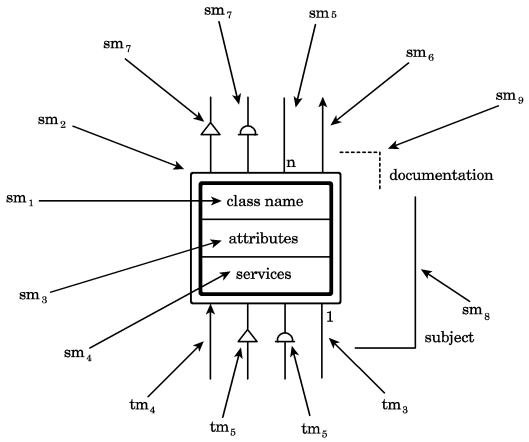
Our project has the indicator values:

CDI ≈ 0,02,
ADI ≈ 0,03,
SDI ≈ 0,06.

How can we measure the OOA/OOD model itself? The main steps in the object-oriented analysis are

1. Finding classes and objects,
2. Identifying structures (class structure (Gen-Spec) and Whole-Part structure),
3. Identifying subjects (as ``view'' of the class structure defined in framed areas),
4. Defining attributes of the classes and the object connections,
5. Defining services of the classes and the message connections.

The documentation contains *all* information that cannot be presented in the drawing. The drawings (also possible in some variants) and the documentation achieve the OOA model. The OOA model must be ``open'' for the measurement. This is given because the OOSE CASE tool - the ObjecTool - is based on a



file set for the graphical models. So, the **measurement tool** OOM (Papritz, 1993) was implemented to measure the OOA model. The evaluation of the OOA step prove the missing inheritance documentation and the small critique that is only directed to an object/class symbol. Further, the estimation of effort, costs and quality is not possible in this development phase (a general problem in the OOSE). The OOD step ensures a full cintinuity to the OOA step. The development phases in OOD are

1. Designing the problem domain component (to achieve further requirements and special aspect of the programming system),
2. Designing the human interaction component,
3. Designing the task management component,
4. Designing the data management component

But, the basis model in the maintenance phase is the OOD model. So, we do not have a method independent specification. We have ''implemented'' 38 classes of 114 classes in the OOD model in the realization of the software measurement laboratory in the design phase as organizatorial orders. So, we have the ***design indicators as***

class modification indicator (CMI) *as*

$$\frac{\text{number of organizational classes}}{\text{number of all designed classes}}$$

attribute modification indicator (AMI) *as*

$$\frac{\text{number of organzational attributes}}{\text{number of all designed attributes}}$$

service modification indicator (SMI) *as*

$$\frac{\text{number of organizational services}}{\text{number of all designed services}}$$

Our project has the values for these indicators

CMI ≈ 0,33,
AMI ≈ 0,48,
SMI ≈ 0,21.

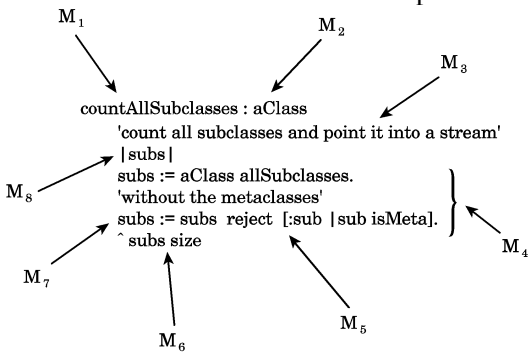
The OOD phase does also missing the relation to the object-oriented implementation (programming) system. So, some browsing activities are necessary in the OOP systerm in the OOD phase. Therefore, we have implemented the OOC tool for browsing in the Smalltalk class library (/Lubahn 94/).

How can we measure the OOP system?

The development steps in the OOP phase include

1. Implementation of ''model" as main object (under the root class in the object-otiented programming system),
2. Implementation of the concrete model,
3. Extension of the object-oriented system with new classes/objects,
4. Extension the object-oriented system with the new methods (as class methods and instance methods; with class variables and instance variables),
5. Modification of existing classes or methods,
6. Testing of the object-oriented application with the designed scenarios.

Here we must choose a special OOP system or a OOP language. The ObjecTool is developed for C++ or Smalltalk implementations. The evaluation of this phase in-



dicates that the OOP ⇒ OOD direction is not possible here. So we introduce maintenance problems at the beginning. The knowledge of the existing OOP systems or libraries is the main effort for an efficient OOSE.

Note that the measures in this development phase would be added by the code measures. For the quality measurement of the process we use the **development complexity** (see /Dumke et al 94/) as set of the used methods and tools and their structure. Other measures (performance etc.) have not been included in this first approach for the development complexity evaluation. The measurement tools were implemented in the same method and programming language to reduce this development complexity. We have implemented a C++ measurement tool (/Kuhrau 94/) in C++ and a Smalltalk measurement extension (/Heckendorf 95/). We implemented 36 classes (as metrics definitions) of the 116 classes in the OOD model that incudes the metrics set. So, we have as ***implementation indicators***

class implementation indicator (CII) *as*

$$\frac{\textit{number of new implemented classes}}{\textit{number of designed classes}}$$

attribute implementation indicator (AII) *as*

$$\frac{\textit{number of new implemented attributes}}{\textit{number of designed attributes}}$$

service implementation indicator (SII) *as*

$$\frac{\textit{number of new implemented services}}{\textit{number of designed services}}$$

These indictaors for our project give the values

$$CII \approx 0{,}31,$$
$$AII \approx 0{,}51,$$
$$SII \approx 0{,}22.$$

The given description of the process measurement is a good example for the method understanding. We can see the essential approach to analyse measurement

in the direction of the μ, m, and M measure mutation.
Some missing tools for the completion of an measurable OOSE method on this basis was designed and implemented.

2.3. The Product Measurement

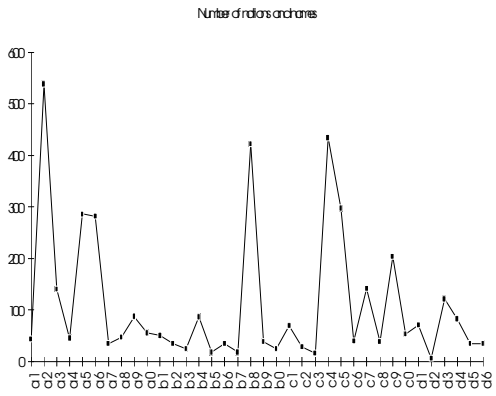
The defined measures for the problem defintion (as html document set) are

- length of the document title ($\sigma\mu_1$),
- average length of all headlines ($\sigma\mu_2$),
- length of the document in Bytes ($\sigma\mu_3$),
- count of words in the document ($\sigma\mu_4$),
- average length of words ($\sigma\mu_5$),
- maximal length of words ($\sigma\mu_6$),
- number of bold or italic words ($\sigma\mu_7$),
- number of notions, names or titles ($\sigma\mu_8$),
- number of dates (times and events) ($\sigma\mu_9$),
- number of lists (UL, OL, DL) ($\sigma\mu_{10}$),

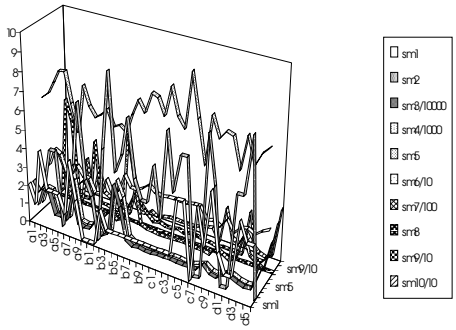
and the structure based measures as

- average number of words in the lists ($\theta\mu_1$),
- number of HR lines ($\theta\mu_2$),
- average number of list elements (in UL or OL) ($\theta\mu_3$),
- average number of list elements in DL ($\theta\mu_4$),
- maximal number of the depth of the lists ($\theta\mu_5$),
- number of hypertext relations ($\theta\mu_6$),
- average length of the loaded files ($\theta\mu_7$),
- average distance of the dates to the actual date ($\theta\mu_8$).

Further measures are the summarized number about the whole problem defintion. The most of these measures are ratio scaled. An implementation of a measurement tool to measure the problem definition (PDM) was necessary (/Foltin 95/). The measurement values for the SMLAB were presented with the EXCEL tool. Three examples of the measurement value presentation are the number of notions in all parts of the problem defintion document set[1]

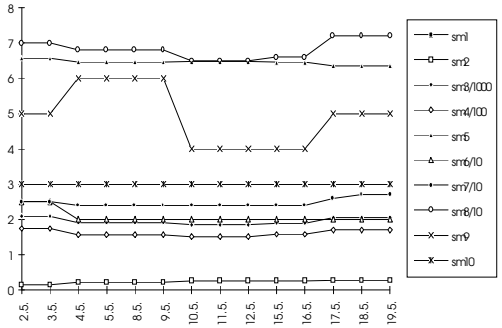


the chosen measures for all documents of the problem definition (as an example of the limited facilities of the EXCEL tool for the presentation of the full details)



and the measurement values of one document about the experiment duration of three weeks. It is good to see the fiew changement in the measure period. This was the reason to define the future measurement activities in a monthly form. The measures give also an overview about the synchro-nized activities of our team.

[1]In these figures the measures $\sigma\mu_i$ was substi-tute with the names sm_i.



The evaluation of the product quality in every development phase is defined as (see also /ISO 9126/) comprehensibility, clarity and usability for the problem statement on the basis of the measures use frequency, availability, size and structure. The correlations between the empirical measures and the indicators are

- **readability:** $\sigma\mu_3$, $\sigma\mu_5$, $\sigma\mu_7$, $\sigma\mu_8$, $\sigma\mu_{10}$, $\theta\mu_1$, $\theta\mu_2$, $\theta\mu_3$, $\theta\mu_5$,
- **mnemonics:** $\sigma\mu_1$, $\sigma\mu_2$, $\sigma\mu_6$, $\theta\mu_4$,
- **clarity:** $\sigma\mu_8$, $\sigma\mu_9$,
- **correctness:** $\theta\mu_8$,
- **useability:** $\theta\mu_6$, $\theta\mu_7$,
- **layout:** $\sigma\mu_7$, $\theta\mu_2$.

The ''measurement'' of the empirical aspects was made by an expertise of the team members as an evaluation in an order scale from ''low'' (1) to ''high'' (5). The results of the empirical evaluation are

criteria	value
readability	3
mnemonics	4
clarity	3.75
correctness	4
useability	4,75
layout	4

Unfortunately, we couldn't prove these correlations in the measurement phase of three weeks in a convinced manner - but also not the opposite fact.

The measures (indicators) for the OOA/OOD model are

- number of abstract classes (sm_1),
- number of object classes (sm_2),
- total number of attributes (sm_3),
- total number of services (sm_4),
- number of object connections (sm_5),
- number of message connections (sm_6),
- number of subclasses (sm_7),
- number of subjects (sm_8),

and the structure based measures

- average number of attributes per class (tm_1)
- average number of services (tm_2),
- average number of object connections (tm_3),
- average number of message connections (tm_4)
- maximum depth of the inheritance (tm_5).

The main measurement values for the OOA model are

measure	value	measure	value
sm1	1	*sm4*	63
sm2	76	*tm1*	2,57
sm3	195	*tm2*	0,83

All measures - excluded the tm_5 - are ratio scaled.

The empirical evaluation of the OOA/OOD model was founded on the

- **completenes,**
- **conformity** and
- **feasability**

for the OOA/OOD phase on the basis of the measures consistency, performance, size and structure. The results of the evaluation are

criteria	value
completeness	2
conformity	2
feasability	3
consistency	3,5
performance	1

A higher granularity of the OOA/OOD model measures is necessary to demonstrate a correlation between the empirical aspects and the model related measures.

In the phase of the OOP we must add the code measures and the other charactersitics of the chosen OO programming system. Such measures are

- the coupling between objects (classes) (M7),
- the lines of method code (M4),
- the method name length (M1),
- the number of method parameters (M2),
- the number of local variables (M8),
- the number of method returns (M6),
- the number of comments (M3),
- the method complexity (M5),

and some other indicators (see /Dumke et al 94/). The measures for the first implmentations as an average manner are (see /Heckendorf 95/ and /Kuhrau 94/)

measure	MPP	Smalltalk
M1	9	8
M2	4	1
M3	3	1
M4	18	12
M5	5	3
M6	1	1
M7	2.8	22
M8	3.2	2.6

The empirical evaluation of the OOP components are founded on the

- understandability,
- stability and
- effort

for the OOP phase on the basis of measuring testability, size, structure and reusability.
In a first approximation of the quality assurance, we can ***want to hold the given quality*** of the OO programming system. So, we must look to the evaluation of the resources.

The values in the table above satisfy this condition.
The most of these measure based on a ordinal scale and can be used only for a classification of the quality.

2.4. The Resource Measurement

The essential aspect in the OOSE is the initial measure of the chosen resources (CASE tools, measurement tools, programming environment etc.).

measure	ObjecTool	OOM
tm3	2	4
tm4	3	0.8
words	8	14
tm8	3	3

tm1	25	2.6
tm2	1.9	1.4

''words'' includes the average number of words in a documentation part and was added to rhe measures to evaluate the documentation level

.

The size of a documentation of all OOA/ OOD model parts is essential for the size or complexity of the implemented class.

The given initial quality can be proved by the given measure values. The tabular on the next page includes someone.

In accordance with our validation aspect we can quantitatively evaluate the usefulness of the chosen object-oriented programming system. For example, we can see the functional approach characteristics in the Smalltalk/V for Windows or in Borland C++ etc. and we can expect a lot of maintenance effort.

measure	**Smalltalk/V**	**ST for Windows**	**Objectworks**	**Borland C++**
# classes	100	170	397	407
depth of the inheritance	5	7	8	6
width of the inheritance	69	118	82	208
average number of class methods	2.7	2.9	2.4	16.3
average number instance methods	17.2	27.7	17.7	0.46
average number of subclasses	1	1.3	6.8	0.74

3. Conclusions

This short paper describes only the main ideas in our present project. This project includes a tool-based evaluation of the object-oriented software development for the methodology of Coad/Yourdon. The goal is to help to quantify the development documents at the beginning for better understanding the OO method and better comparing this method with the other OO development paradigms.

References

/Abreu et al 94/ Abreu, F.B.; Carapuca, R.: *Candidate Metrics for Object-Oriented Software within a Taxonomy Framework.* Journal of Systems and Software, 26(1994), pp. 87-96

/Arora et al 95/ Arora, V.; Kalaichelvan, K.; Goel, N.; Munikoti, R.: *Measuring High-Level Design Complexity of Real-Time Object-Oriented Systems.* Proc. of the Annual Oregon Workshop on Software Metrics, June 5-7, 1995, Silver Fall, Oregon, pp. 2/2-1 - 2/2-11

/Binder 94/ Binder, R.V.: *Design for Testatbility in Object-Oriented Systems*. Comm. of the ACM, 37(1994)9, pp. 87-101

/Cant et al 94/ Cant, S.N.; Henderson-Sellers, B.; Jeffery, D.R.*: Application of cognitive complexity metrics to object-oriented programs.* Journal of Object-Oriented Programming, July-August 1994, pp. 52-63

/Chidamber et al 94/ Chidamber, S.R.; Kemerer, C.F.: *A Metrics Suite for*

Object-Oriented Design. IEEE Transations on Software Engineering, 20(1994)6, pp. 476-493

/Churcher et al 95/ Churcher, N.I.; Shepperd, M.J.: *Towards a Conceptual Framework for Object Oriented Software Metrics*. Software Engineering Notes, 20(1995)2, pp. 68-75

/Coad et al 93/ Coad, P,; Nicola, J.: *Object-Oriented Programming*. Prentice-Hall Inc., 1993

/Dumke et al 94/ Dumke, R.; Kuhrau, I.: *Tool-Based Quality Management in Object-Oriented Software Development.* Proc. of the Third Symposium on Assessment of Quality Software Development Tools, Washington D.C., June 7-9, 1994, pp. 148-160

/Dvorak 94/ Dvorak, J.: *Conceptual Entropy and its Effect on Class Hierarchy*. IEEE Computer, June 1994, pp. 59-63

/Fenton 91/ Fenton, N.: *Software Metrics - A rigorous approach*. Chapman & Hall Publ., 1991

/Foltin 95/ Foltin, E.: *Implementation of a problem definition measurement tool PDM*. Technical Report, University Magdeburg, 1995

/Heckendorf 95/ Heckendorf, R.: *Design and Implementation of a Smalltalk Measurement Extension*. Technical Report, University of Magdburg, 1995

/IEEE 93/ IEEE Standard for a *Software Quality Metrics Methodology*. IEEE Publisher, March 1993

/ISO9126/ ISO/IEC 9126 Standard for Information Technology, *Software Product Evaluation - Quality Charateristics and Guidelines for their Use.* Geneve 1991

/John et al 95/ John, R.; Chen, Z.; Oman, P.: *Stactic Techniques for Measuring Code Reusability*. Proc. of the Annual Oregon Workshop on Software Metrics, June 5-7, 1995, Silver Fall, Oregon, pp. 3/2-1 - 3/2-26

/Jones 94/ Jones, C.: *Gaps in the object-oriented paradigm*. IEEE Computer, June 1994, pp. 90-91

/Karunanithi, S.; Bieman, J.M.: *Candidate Reuse Metrics for Object Oriented and Ada Software*. Proc. of the Firts Int. Software Metrics Symposium, May 21-22, Baltimore, 1993, pp. 120-28

/Kuhrau 94/ Kuhrau, I.: *Design and Implementation of a C++ Measurement Tool*. Master Thesis, University of Magdeburg, March 1994

/Lee et al 94/ Lee, A.; Pennington, N.: *The effects of paradigm on cognitive activities in design.* Int. Journal of Human-Computer Studies (1994)40, pp. 577-601

/Lejter et al 92/ Lejter, M.; Meyers, S.; Reiss, S.P.: *Support for Maintaining Object-Oriented Programs.* IEEE Transactions on Software Engineering, 18(1992), pp. 1045-1052

/Li et al 95/ Li, W.; Henry, S.; Kafura, D.; Schulman, R.: *Measuring object-oriented design.* JOOP, July-August 1995, pp. 48-55

/Lorenz 93/ Lorenz, M.: *Object-Oriented Software Development.* Prentice Hall Inc., 1993

/Lubahn 94/ Lubahn, D.: *The OOC tool description*. Technical Report, University of Magdeburg, 1994

/Ott et al 95/ Ott, L.M.; Kang, B.; Bieman, J.M.; Mehra, B.: *Developing Measures of Class Cohesion for Object-Oriented Sofwtare*. Proc. of the Annual Oregon Workshop on Software Metrics, June 5-7, 1995, Silver Fall, Oregon, pp. 3/1-1 - 3/1-11

/Papritz 93/ Papritz, T.: *Implementation of an OOM tool for the OOA model measurement*. Technical Report, University of Magdeburg, July 1993

/Rocache 89/ Rocache, D.: *Smalltalk Measure Analysis Manual*. ESPRIT Project 1257, CRIL, Rennes, Franche, 1989

/Szabo, R.M.; Khoshgoftaar, T.M.: *Modeling Software Quality in an Object Oriented Software System.* Proc. of the Annual Oregon Workshop on Software Metrics, June 5-7, 1995, Silver Fall, Oregon, pp. 2/3-1 - 2/3-20

/Zuse 94/ Zuse, H.: *Foundations of the Validation of Object-Oriented Software Measures*. in: Dumke/Zuse: Theory and Practice of Software Measurement. Deutscher Universitätsverlag, Wiesbaden, 1994, pp. 136-214