

# 12<sup>th</sup> International Workshop on Software Measurement

## IWSM 2002

### Workshop Program

October 7 – 9, 2002, Magdeburg, Germany

*Venue: Computer Science Building 29, University of Magdeburg*

#### Monday, October 7, 2002:

8.00:            *Workshop and Tutorial **Registration**, Room 319*

9.00 – 13.00:   **Tutorials**

***Charles Symons**, Software Measurement Services Ltd., London, UK:  
COSMIC FFP – for Sizing Business and Real-time Software,  
Room K058*

***Alain Abran**, Ecole de Technologie Superieure, Montreal, Canada:  
Concept and Practical Use of the ISBSG Portal,  
Room 335*

***Horst Zuse**, Technical University Berlin, Germany:  
Metrics in the Software Life Cycle,  
Room E037*

13.00 – 14.30:   *Lunch*

14.30 – 15.30:   *Community session of the **GI FG 2.1.10** members  
Room 335*

15.30 – 17.00:   *Community session of the **COSMIC** members  
Room E037*

17.00 – 18.30:   *Community session of the **DASMA** members  
Room K058*

18.30:            *Exhibitors' Reception*

**Tuesday, October 8, 2002:***Venue: Computer Science Building 29, Room 307*8.45 – 9.00: Alain Abran, Reiner Dumke: *Welcome and Introduction***Chair:** *Reiner Dumke, University of Magdeburg*9.00 – 9.45: **Keynote Horst Zuse** (TU Berlin, Germany):  
*Problems and Pitfalls in Software Metrics Applications*9.45 – 10.15: Alain Abran, Asma Sellami (ETS Montreal, Canada):  
*Models of the Measurement Concepts in the ISO Vocabulary of Terms in Metrology*10.15 – 10.45: Harry Sneed (SDS Vienna & CaseConsult Wiesbaden, Germany):  
*XMI-Relational Model of a Software Metric Database*

10.45 – 11.00: Coffee Break

**Chair:** *Alain Abran, Ecole de Technologie Superieur Montreal*11.00 – 11.30: Alain Abran, Peter Fagg, Roberto Meli, Charles Symons (COSMIC Consortium):  
*Clarifications of the COSMIC FFP Method of Functional Sizing*11.30 – 12.00: Malcolm Jenner (University of Woverhampton, UK):  
*Automation of Counting of Functional Size Using COSMIC FFP in UML*12.00 – 12.30: Jean-Marc Desharnais, Tim Küssing, Alain Abran; Andre Mayers (University of Quebec & University of Nuremberg & ETS & University of Sherbrook, Canada):  
*Design of a Diagnostic Tool to Improve the Quality of the Functional Measurement*

12.30 – 14.00: Lunch

**Chair:** *Charles Symons, Software Measurement Service Ltd., UK*14.00 – 14.30: Maya Daneva (TELUS Mobility, Canada):  
*Comparing ERP Requirements Engineering Processes: a Case Study*14.30 – 15.00: Mathias Lothar, Andreas Schmietendorf, Thomas Böhm, Reiner Dumke (University of Magdeburg):  
*Quality Evaluation of Large Scale Software Systems*

15.00 – 15.30: Roland Neumann, Lars Grunske (Institute of Hasso-Plattner, Potsdam, Germany):  
*Hierarchical Software Quality Models – a Step Towards Quantifying Nonfunctional Properties*

15.30 – 15.45: Coffe Break

**Chair:** Harry Sneed, CaseConsult Wiesbaden, Germany

15.45 – 16.15: Manfred Bundschuh (AXA Cologne, Germany):  
*Estimation of Maintenance Tasks*

16.15 – 16.45: Andreas Schmietendorf, Reiner Dumke (T-Systems Berlin & University of Magdeburg, Germany):  
*Metrics-based Analysis of Enterprise Java Beans Components*

16.45 – 17.15: Luigi Buglione, Alain Abran (SchlumbergerSema Rome, Italy & ETS Montreal, Canada):  
*ICEBERG: a Different Look at Software Project Management*

19.30: Social Event

**Wednesday, October 9, 2002:**

*Venue: Computer Science Building 29, Room 307*

**Chair:** Horst Zuse, TU Berlin

9.00 – 9.45: **Keynote Alain Abran** (Ecole Technologie de Superieur, Montreal):  
*The SWEBOK Initiative and Software Measurement Intentions*

9.45 – 10.15: Stefan Jungmayr (FernUniversität Hagen, Germany):  
*Testability Measurement and Software Dependencies*

10.15 - 10.45: Cornelius Wille, Reiner Dumke, Stanimor Stojanov (University of Magdeburg, Germany & University of Plovdiv, Bulgaria):  
*New Measurement Intentions in Agent-based Systems Development and Application*

10.45 - 11.00: Coffe Break

**Chair:** Manfred Bundschuh, AXA Cologne & DASMA e. V.

11.00 - 11.30: Antonia Bertolino, Gaetano Lombardi, Eda Marchetti, Reffaela Mirandola (CNR Pisa & Rome University & EricssonLab Rome, Italy):  
*Software Performance Measures to Assist Decision Makers within the Rational Unified Process*

- 11.30 - 12.00: Alain Abran, Reiner Dumke, Jean-Marc Desharnais, Iphigenie Ndyaje, Christian Kolbe (ETS Montreal, Canada & University of Magdeburg, Germany):  
*A Strategy for a Credible & Auditable Estimation Process Using the ISBSG International Data Repository*
- 12.00 - 12.30: Robert Hürten (Hürten & Partner Consulting, Blankenheim, Germany):  
*Why does the Function Point Analysis Find so Little Acceptance?*
- 12.30 – 14.00: Lunch
- Chair:** Günter Büren, Büren & Partner Nuremberg, Germany
- 14.00 – 14.30: Marek Leszak, Werner Brunck, Gerd Moessler (Lucent Technologies, Nuremberg, Germany):  
*Analysis of Software Defects in a Large Evolutionary Telecommunication System*
- 14.30 – 15.00: Hannu Toivonen (Nokia Research Center, Finland):  
*Defining Measures for Memory Efficiency of the Software in Mobile Terminals*
- 15.00 – 15.30: Reiner Dumke, Mathias Lothar, Cornelius Wille (University of Magdeburg):  
*Situation and Trends in Software Measurement – A Statistical Analysis of the SML@b Metrics Bibliography*
- 15.30: Reiner Dumke: *Final Discussion and Closing*

**FUNCTION POINT PROGNOSIS APPROVED**  
**Regression Analysis for Approximation of**  
**Function Point Counts**

*Dipl.-Math. Manfred Bundschuh*  
AXA Service AG, Cologne, President of DASMA e.V.

## 1 Environment

In 2001 the AXA Service AG in Cologne completed the total counts of all 78 application systems (without SAP applications) with totally about 100,000 unadjusted Function Points (FP's). Actually more than 100 staff members are trained in FP counts and project estimation in order that in each group of developers there is at least one FP and estimation „expert“. The competence center consists of 3 people who are also responsible for the multiproject management and project management tools (Business Engine and MS Project). The Business Engine administers about 250 projects with 20,000 assignments of 1,100 people delivering timesheets for weekly planning and reports as well as monthly accounting.

### **Function Points mean throughout this paper IFPUG 4.0 unadjusted FP's.**

FP counts are in AXA Service AG obligatory at least at the end of the requirements analysis and at project post mortem. Function Point prognosis, as described in this paper, instead of FP count is obligatory during the feasibility study and at project start. The counts and their details are documented centrally in the estimation competence center and in the Function Point Workbench and Excel charts (see chapter Project Register Database) and thus could be investigated for gaining several metrics and especially for early estimation purposes.

AXA Service AG is the outsourced noninsurance part of AXA Insurances in Germany. The IT Department includes about 650 IT Professionals with approximately 50 Project leaders. Besides there is the outsourced computing center with about 250 staff members. The insurance branches deliver about 160 IT-Coordinators supporting the IT-Projects. IT-development is mostly host-based with COBOL programming. There exist about 2,700 Databases (1,600 CICS, 1,100 IMS) and 2,500 DB2 tables (1,600 production, 900 disposition), about 7,800,000 transactions per day (5,600,000 CICS and 2,200,000 IMS). PC Projects use Optima++, a C++ Shell, for programming and Internet Programming is done with Java.

IT-Projects in AXA Service AG do mostly develop interactive database administration systems for e.g. car or life insurance or claims management, within a very complex environment with centralized databases for e.g. insurance partners, document management etc.

## 2 Motivation

Project estimations are required as early as possible – not only from the contractors but also from every project leader. Because of the importance of early estimation methods Roberto

Meli and Luca Santillo [MS 99] published a comparative overview of Function Point estimation methods which shows a valuable collection of worldwide efforts in this direction. Since Function Point counts are based on the requirements documentation so called Function Point prognoses or approximations proved to be helpful in praxi to aid early estimations.

Our experience shows, that the necessary informations for such approximations can be gained very early, at the beginning of a project (or in stable environments even before) in discussions with the project leader. In a few cases we did use this Function Point Prognosis a year before project start, of course with adding large percentages for error, uncertainty, early estimation and risk.

We collect the informations about interfaces and parts of the project together with the Competence Center staff and document it together with a diagram of the application boundary which is also used for our architecture atlas and together with a so called counting log which is a simple Word document giving important informations about special decisions concerning the the FP counts. Since we developed a Function Point Prognosis (see FESMA Conferences 1998 and 1999) we approved this method for early estimation by a larger base of Function Point Counts in 2001. The databases of the 1998 and 1999 investigations are published in [BU 98] and [BU 99].

### 3 Project Register Database

We used our project register database (EXCEL) which shows detailed informations (extracted from the Function Point Workbench) for each FP count, giving the quantity as well as the Function Points of EI's, EO's, EQ's, ILF's and EIF's for each of this components and some informations more (platform, VAF, adjusted Function Points). If a project was counted repeatedly only the most actual count is shown and older counts are kept in a history table. There are sums of the quantity of EI's and EO's, of ILF's and EIF's which were needed for our research.

#### **IO means throughtout this paper the sum of the number (quantities) of EI and EO.**

The idea for this research was to find out if there were hidden informations in that data collection. The results proved to be very productive.

In my first presentation<sup>1</sup> on the FESMA Conference 1998 in Antwerp [BU 98] I presented the results based on about 20 counts which increased to 39 counts a year later. Since there was afterwards a break of 2 years in Function Point counting because the competence center staff was involved in introducing a multiproject management tool with time accounting, the total counts of all 78 application systems (AS) could not be accomplished until end of 2001.

This goal could only be reached with one of 5 Balanced Scorecards for the department leaders, combining their success in counting all of their application systems with 20 % of their financial bonus (this was an essential management support for the success of our metrics

---

<sup>1</sup> M. Bundschuh, Function Point Prognosis, FESMA 98 „Business Improvement through Software Measurement“, Antwerp, Belgium, May 6-8, 1998, pp. 463 - 472

programme). Hence we were curious if the new results would be consistent and if parts of the project data base would deliver confident findings.

4 Investigations

Besides the investigation of the formulae for prognosis we investigated also the Function Point proportions and average function complexity of our Function Point counts and compared it e.g. with the ISBSG (International Software Benchmarking Standards Group) Data. The ISBSG ([www.isbsg.org.au](http://www.isbsg.org.au)) publishes every year a collection of metrics (The Benchmark, it can be ordered in Europe via the DASMA e.V., [www.dasma.de](http://www.dasma.de)), in the actual release based on more than 1,300 projects worldwide. Since we did this research several times during the last years we are lucky to have now at least 3 historical annual metrics of our own data for comparison, as can be seen in the following chapters.

4.1 Value Adjustment Factor

One of the first results of our data collection was the perception that the VAF (Value Adjustment Factor) of our counts is typically in the range of 0.73 – 1.22, with an average of 0.95 in the 2001 data (0.93 in 1998) and an average of 0.94 for Host and 0.96 for PC environment. The average for migrations is 0.73. We use this metrics for quality assurance of our Function Point counts since 1998.

4.2 Function Component Proportions

The following Fig. 1 shows the historic development of the function component proportions in AXA Service AG. Of course, the first two years are not very representative. The figures from 1998 and 2001 are similiar and the division into Host and PC development shows differences which should be carefully observed in future. The domination of the EI's and EO's (61 % together) seems to be the reason for the strong correlation between IO's and the unadjusted FP's - the main result of this research.

2001	Number of Application Systems	Percent of Function Points				
		EI	EO	EQ	ILF	EIF
<b>Total</b>	78	22	39	8	16	14
<b>Host</b>	69	21	40	8	16	15
<b>PC</b>	9	28	31	12	19	10
<b>ISBSG Rel. 6</b>	238 new development projects	33.5	23.5	16	22	5
<b>Metricviews</b>		26-39	22-24	12-14	24	4-12
<b>Checkpoint</b>		20	24	10	43	3

<b>1998 Total</b>	39	25	39	14	17	6
<b>1996/7 Total</b>	20	27	39	11	18	5
<b>1997 Total</b>	12	18	43	12	18	9

<b>1996 Total</b>	8	34	35	11	18	2
-------------------	---	----	----	----	----	---

**Fig. 1: Function Component Proportions**

*It can easily be seen, that EO's dominate in AXA Service AG ( 39% ) compared with Morris' and Desharnais' publication [MD 96] (22%-24%), and the quick estimation mode of Checkpoint, the estimation tool from Capers Jones' firm SPR (Software Productivity Research) in Burlington, MA, (20 %) whereas ILF's are of minor importance (16% vs. 24%, 43%, resp. ). Because of this peculiarity one conclusion was not to use Checkpoint ( EIF + ILF = 46% vs. 23 % in AXA Service AG) in Quick Estimate mode for the estimation of FP's.*

*The reason for the major importance of EO's may be that AXA Service AG has many centralized management information.*

*In 2000 we accomplished an error calculation with the 1998 data by using the percentage of each component to calculate 100 % from it and compared the result with the actual Function Point count. The findings are published in [BU 00] and showed errors ranging from 37 % (EO's) to 48 % (EQ's). Hence we do use the percentages of the components only as a rule of thumb for the quality assurance of our Function Point counts.*

#### 4.3 Average Function Complexity

We used the EXCEL problem solver to calculate from the project register data base the average function complexity of the five components, i.e. how many FP's a „typical“ EI, EO, EQ, ILF, EIF has in our environment. It is widely agreed that this measure is stable and can be used as a rule of thumb for quick estimation of counts, since the components then need not be classified as low, average or high. SPR Function Points e.g. use the average IFPUG classification for Function Point estimation.

<b>2001</b>		<b>Average Function Points</b>				
<b>Platform</b>	<b>Number of Application Systems</b>	<b>EI</b>	<b>EO</b>	<b>EQ</b>	<b>ILF</b>	<b>EIF</b>
<b>Total</b>	78	4.7	5.9	4.4	8.6	6.5
<b>Host</b>	69	4.7	5.9	4.6	8.7	6.5
<b>PC</b>	9	4.3	5.7	3.8	7.6	6.5
<b>IFPUG</b>		4	5	4	10	7
<b>ISBSG</b>	Release 5	4.3	5.4	3.8	7.4	5.5
	Release 5 Europe	4.2	4.9	3.8	7.2	5.3

<b>1998</b>	<b>Number of Application Systems</b>	<b>EI</b>	<b>EO</b>	<b>EQ</b>	<b>ILF</b>	<b>EIF</b>
<b>Total</b>	39	4.6	5.7	4.3	8.2	6.1
<b>Host</b>	28	4.8	5.7	4.5	8.5	6.2
<b>PC</b>	11	4.0	5.7	3.9	7.3	5.4

<b>1997</b>	<b>Number of Application Systems</b>	<b>EI</b>	<b>EO</b>	<b>EQ</b>	<b>ILF</b>	<b>EIF</b>
-------------	--------------------------------------	-----------	-----------	-----------	------------	------------



<b>Total</b>	<b>20</b>	<b>4.6</b>	<b>5.5</b>	<b>4.3</b>	<b>8.1</b>	<b>5.7</b>
--------------	-----------	------------	------------	------------	------------	------------

**Fig. 2: Average Function Complexity**

*It can easily be seen from Fig. 2, that the average Function Points increased in time, which may be caused by growing complexity in application development environment.*

In 1998 we tested the applicability of this typical FP's for estimation purposes by multiplying it with the quantities of the EI's, EO's, EQ's, ILF's, EIF's, resp. and compared the results with the unadjusted Function Points of the counts. The error was less than 26 %.

**4.4 Function Point Ratios**

*One would expect 3 inputs (add, change, delete) at least, 1 output and 1 EQ for maintenance of a file. The following results show the averages in AXA Service AG.*

**4.4.1 Ratios of Components**

*There are remarkable differences between the before mentioned expectations and also some differences between the ratios in our application systems (AS) and the ISBSG findings.*

Application Systems	AXA Service AG			ISBSG Rel. 5	
	2001	1998	1997	Europe	Total
Quantity	<b>78</b>	<b>39</b>	<b>20</b>	<b>32</b>	<b>238</b>
EI per ILF	<b>2.6</b>	<b>2.7</b>	<b>2.7</b>	<b>3.8</b>	<b>2.9</b>
EO per ILF	<b>3.6</b>	<b>3.3</b>	<b>3.7</b>	<b>2.6</b>	<b>1.5</b>
EQ per ILF	<b>0.9</b>	<b>1.4</b>	<b>1.2</b>	<b>1.9</b>	<b>1.1</b>
EIF per ILF	<b>0.6</b>	<b>0.5</b>	<b>0.4</b>	-	-

We also calculated the ratios per input as well as the ratios per output.

78 Application Systems	2001	78 Application Systems	2001
EO per EI	<b>1.3</b>	EI per EO	<b>0.7</b>
EQ per EI	<b>0.3</b>	EQ per EO	<b>0.3</b>
ILF per EI	<b>0.4</b>	ILF per EO	<b>0.3</b>
EIF per EI	<b>0.2</b>	EIF per EO	<b>0.2</b>

**Fig. 3: Ratios of Components**

**4.4.2 Ratios of Function Points per Component**

The ratios of Function Points per ILF, input and output were calculated, too.

78 AS	2001	78 AS	2001	78 AS	2001
EI FP's per ILF	<b>12.2</b>	EO FP's per EI	<b>8.0</b>	EI FP's per EO	<b>3.4</b>
EO FP's per ILF	<b>21.0</b>	EQ FP's per EI	<b>1.5</b>	EQ FP's per EO	<b>1.1</b>
EQ FP's per ILF	<b>4.0</b>	ILF FP's per EI	<b>3.3</b>	ILF FP's per EO	<b>2.4</b>

EIF FP's per ILF	<b>4.2</b>	EIF FP's per EI	<b>1.6</b>	EIF FP's per EO	<b>1.2</b>
------------------	------------	-----------------	------------	-----------------	------------

**Fig. 4:** Ratios of Functions Points per Component

5 Function Point Prognosis

Regression analysis on our project register database was used in order to find correlations between the number of the components and the unadjusted Functions points of the counts. The idea for this regression analysis arose while reading John E. Gaffrey's publication [GA 94]. The result of the research was, that the sum of the quantities of EI's and EO's ( IO's in our diction ) is correlated with about  $R^2 \geq 0.95$  ( $R \geq 0.97$ ) to the total amount of FP's of a count and can thus be used as a „rule of thumb“ for the „Prognosis“ of FP's when the FP's of EQ's , ILF's and EIF's are not known.

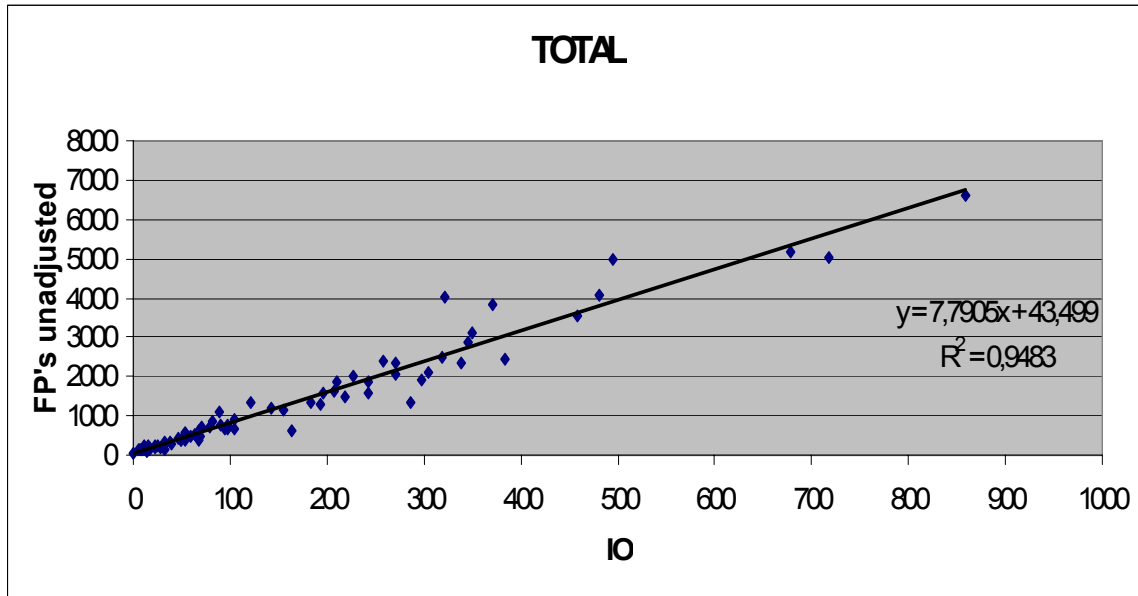
An interesting result was, that the correlation was not as much reliable ( $R^2$  mostly less than 0,9) for other components as well as for data subsets of small, medium and large counts and not better with polynomial regression. Of course the use of FP's instead of the IO's for the prognosis gives a stronger correlation, but the higher effort for classification of the components instead of only counting the inputs and outputs is not adequate for the higher precision. One should always keep in mind, that estimation has to do with uncertainty per se.

The 1998 data were analyzed independantly by Damien Noel [NO 99], a doctorand from Alain Abran, in a joint research with the Software Engineering Management Research Laboratory, Department Informatique, Universite du Quebec a Montreal (UQAM), Canada, who obtained the same results. He applied the same method to 7 projects with COSMIC Full Function Points (FFP's), in order to find a similiar correlation for FFP's, but the sample seemed to be too small for reliable results. He reported in his thesis an error margin of 20 %.

<b>2001</b>	<b>Number of counts</b>	<b>R<sup>2</sup></b>	<b>Error in %</b>	<b>Formula for Prognosis</b>
<b>Total</b>	78	0.9483	13	<b>FP = 7.8 * IO + 43</b>
<b>Host</b>	69	0.9498	12	<b>FP = 7.9 * IO + 40</b>
<b>PC</b>	9	0.9503	21	<b>FP = 6.4 * IO + 172</b>
<b>1998</b>	39	0.9589	20	<b>FP = 7.6 * IO + 50</b>
<b>Host</b>	28	0.9580		<b>FP = 7.9 * IO + 11</b>
<b>PC</b>	11	0.9760		<b>FP = 6.5 * IO + 134</b>
<b>1997</b>	20	0.9525	13 (Median 11)	<b>FP = 7.3 * IO + 56</b>

**Fig. 5: Function Point Prognosis Formulae**

The following Fig. 6 visualizes the regression analysis result for all counts.



**Fig. 6: Regression Analysis Example**

## 6 Conclusions

A good documentation of counting and estimation data is a treasure for metrics programmes. Our investigations show that valuable metrics can be gained from the collected data. A surplus benefit was the finding (via regression analysis) of our prognosis formulae which help us to estimate FP's very early. Since FP's are an important measure for the estimation of effort we thus gained the synergetic benefit to be able to do reliable estimations very early in the lifecycle of our IT projects. Of course a complete count of FP's ad the ende of the requirements analysis is obligatory, as well as an improved estimation ad this time.

We heard from other firms that they did similiar research. There is strong evidence that different environments will lead to other results. Hence each organisation should develop iistown heuristic solutions. Nevertheless comparisons with other metrics are valuable for teh enterprise.

There was a long way to arrive at this results of application counting from the start of the introduction of our metrics programme in 1996. The success could only be achieved with enough management support. The year 2002 is devoted to introduce project FP counting and estimation as well as the introduction or productivity measures.

## References

- [BU 98] M. Bundschuh: Function Point Prognosis, FESMA 98 „Business Improvement through Software Measurement“, Antwerp, Belgium, May 6-8, 1998, pp. 463 – 472
- [BU 99] M. Bundschuh: Function Point Prognosis Revisited, FESMA 99, Amsterdam, Netherlands, October 4-7, 1999, pp. 287-297
- [BU 00] M. Bundschuh: Function Point Approximation with the five Logical Components, FESMA 00, Madrid, Spain, October 18-20, 2000
- [BF 00] M. Bundschuh, A. Fabry: Aufwandschätzung von IT-Projekten, MITP Verlag, Bonn, 2000, ISBN 3-8266-0534-9
- [GA 94] John E. Gaffney, Jr. from Corp. Software Productivity Consortium in Herndon, Virginia 22070 in his Presentation: „A Simplified Function Point Measure“ at the IFPUG 1994 Fall Conference, Oct. 19-21, 1994 in Salt Lake City, Utah
- [MD 96] M. Morris, J. M. Desharnais: „Validation of the Function Point Counts“, in: Metricviews, Summer 1996, p. 30
- [MS 99] Roberto Meli, Luca Santillo: Function Point Estimation Methods: A Comparative Overview, FESMA 99, Amsterdam, The Netherlands, October 4-8, 1999, pp. 271 – 286
- [NO 99] Damien Noel: Analyse statistique pour un design plus simple de la methode de mesure de taille fonctionelle du logiciel dans des contextes homogenes, doctoral thesis, presented to Alain Abran, UQUAM, 2.8.

# QUANTIFICATION OF SURVIVABILITY AT THE CODE LEVEL THROUGH SOFTWARE MEASUREMENT

*Nadine Hanebutte*

University of Idaho, Department of Computer Science, Moscow, ID 83844, USA  
hane@cs.uidaho.edu

**Abstract:** *This paper attempts to discover the possibility of quantifying software survivability at the code level. Furthermore, the framework for a code metric will be established that will allow the estimation of software survivability.*

*The term "survivability" is not formally defined according to the literature and research conducted in this field. While an informal definition allows an intuitive understanding, it does not specify how to quantify observations in a way that two people would agree about the state of survivability of a system.*

*This paper will establish a definition framework and technique for quantification under consideration of commonly used informal definitions, as in [Mead2000] or [Neumann2000].*

*A quantification of survivability on the code level would allow measuring the effect of any changes to the system in terms of their impact on survivability. Additionally, a quantification of survivability will increase the understanding of the term itself and will help to clarify its meaning in the context of other terms like reliability, safety, fault tolerance, intrusion tolerance or security [Voas2002].*

**Keywords:** *Software Measurement, Software Survivability, Source Code Metrics*

## 1 Introduction

Recently there has been recognition that absolute security is unattainable [Dietrich2002]. As a result, a research area has formed known as *survivability*. Its focus is the creation or improvement of systems that are able to continue to deliver services even through security measures have failed.

With a constantly increasing number of discovered vulnerabilities [CERT], systems are to be built that can deal with these possible threats. But any attempt to increase or introduce survivability to a system can not be verified, since there is no clear criteria against which to measure success or progress [Dietrich2002].

This is because there is only an intuition-based agreement on what survivability means. Two of the most commonly used definitions are: "We define survivability as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. We use the term system in the broadest possible sense, including networks and large-scale systems of systems" [Mead2000] and "For the purposes of this report, survivability is the ability of a computer-communication system-based application to satisfy and to continue to satisfy certain critical requirements (e.g., specific requirements for security, reliability, real-time responsiveness, and correctness) in the face of adverse conditions." [Neumann2000]

From these definitions, it is apparent that the cornerstones of a general understanding of survivability are:

- The *something* that has to function

- The threat which wants the *something* to stop functioning
- The fact that the threat is driven by an intelligent force

The first two points above imply a conditional robustness. The third point gives the nature of the conditions, which are to be taken into consideration.

Threats to a system are manifold. Besides classifying them into either intentional or accidental, they can be categorized according to their source fault<sup>2</sup>. These can be logical errors, insufficient implementation of requirements, or "bad" coding

While both definitions allow an intuitive understanding, they do not specify how to measure or state changes in the survivability of a system. This is insufficient.

Without quantification it is not possible to state the survivability of a system unambiguously. Furthermore, there is no way to determine whether any applied method to increase survivability succeeded. Therefore, assessment is one of the key challenges, which can only be solved after a sound definition is found [Voas2002].

The focus of this research is to define and make possible quantification of survivability of the source code in terms of the statements that can make the code vulnerable. These are functions or statements which are badly implemented in the programming language itself. These expressions, when used with certain syntax, or without a check of processed arguments, can yield an exploitable expression. For instance the string copy function *snprintf()* can be used to read from random memory addresses.

## 2 Survivability

According to [Avizienis2001] a general definition of survivability in military standards can be traced back to the late sixties, where it was defined as “a system’s capacity to resist a hostile environment so that it can fulfill its mission” (see e.g., MIL-STD-721 or DOD-D-5000.3).

This definition features a rather important term - hostile. As opposed to reliability, where threats are mostly seen as unintentional, hostility implies that there are not just "natural" causes to be dealt with, like component aging or the fact that a human using software enters an invalid input. Hostility adds an important flavor, which is the fact that when talking of survivability, an intelligent source has to be considered that is creating or taking advantage of faults. Therefore, probabilistic assumptions about the occurrences for certain fault types (as used for systems reliability analysis) are not valid.

Survivability can be defined as:

A property of a system, subsystem, equipment, process or procedure that provides a defined degree of assurance that the named entity will continue to

---

<sup>2</sup> For hardware, a fault can trigger an error. This error can cause a failure. In the context of software, the error is seen as the root event, which results in a fault, and can eventually cause failure. For the purposes of this paper, the terminology from the hardware community will be used even when talking about software.

function during and after a natural or man-made disturbance; e.g. nuclear burst. [US1037C]

The most interesting statement within this definition is the term *degree*. Degree implies that survivability is an attribute, which can be quantified for a specific entity with a value. (The author continues)

Note: For a given application, survivability must be qualified by specifying the range of conditions over which the entity will survive, the minimum acceptable level of post-disturbance functionality, and the maximum acceptable outage duration.

There is another point to be considered: the condition or the environment under which any statement about the observed entity must hold. This can be the expected operational environment as well as the possible threats. Another measure of system dependability is *availability* as a measure of the current state of a system.

Availability  $A(t)$  is a function of time, defined as the probability that a system is operating correctly and is available to perform its functions at the instant of time  $t$ . Availability differs from reliability in that reliability depends on the interval of time, whereas availability is taken at an instant of time. [Johnson1989]

If the survivability of a system is seen as a constant, there has to be another measure to account for the system at any specific point in time during run-time and therefore under any threat. This run-time measure corresponds to survivability just like availability does to reliability. For the purpose of this paper it will be called *dynamic survivability* to differ from survivability which is here seen as a static measure. Another problem to be considered is the operational profile of a system as it influences the overall survivability as well.

Survivability is not all or nothing. There are different degrees to which you can achieve it. ... Survivability is highly context sensitive and environment-sensitive. Thus for system A, the survivability of A in environment B may or may not equal the survivability of A in environment C [Voas2002].

Besides a measure of survivability on the program or system level, there has to be a way to combine the measures of interacting systems into a composite survivability, just as is done for reliability. Constructs like fault trees or reliability block diagrams allow modeling of system reliability.

There are basically four cornerstones that can be found in the aforementioned definitions of survivability:

1. The entity : The *something* whose attributes are to be measured
2. The degree : The measured value
3. The context : The condition in terms of the given environment
4. The threats : Driven by an intelligent force



This four-tuple is seen in other research areas that are closely related to survivability, as well. This sometimes leads to mistakes in differentiating the research fields and their medium. For example, the ability of a system to recover after a threat or to deal with an intrusion is often considered as a part of the security measures. However, security is defined as the absence of unauthorized access to, or handling of, system-state [Avizienis2001] and therefore merely focuses on the detection of intrusions only.

The reason why survivability is so closely aligned to the area of security can be seen in the fact that the intelligent adversary is most difficult environmental threat to which to respond or survive. The intelligent adversary targets primarily the traditional security attributes: confidentiality, availability and integrity [Longstaff2002].

## 2.1 Decomposition of System Survivability

In [Mead2000] the survivability is described as the capability of a system to fulfill a mission, where mission is seen as a higher goal, objective or a set of high-level requirements. The term *system* is used in a broad sense including networks and system of systems.

This research project attempts to assess survivability by decomposing a system into its sub-components. Therefore the smallest components that make up a system are to be identified. The system survivability than will consist of a composite survivability of all its components.

On the software level, each system can be decomposed into a set of operations. Executing system functionalities performs these operations. Finally, each functionality is implemented in a set of software modules [Elbaum1999]. A *module* here is equivalent to a function in, for example, the C programming language. The module is considered the smallest measurable unit.

The overall system survivability is therefore a composite of all modules required for the system to fulfill its mission. A mission can only be fulfilled if this minimum set of software modules is functioning. Depending on the system composition, there can be more than one set of modules that allows mission fulfillment. These sets can be partially overlapping as well as disjoint.

Mission fulfillment fails if none of these minimum sets is complete during a threat or if at least one module from each set is compromised. For instance if there is one module that is part of each minimum set, and this module is compromised, the mission will fail.

In both the mission-centric as well as the system-centric view of survivability, the module can be seen as the smallest measurable unit.

Survivability measured on the code level is based on the fact that, given a specific language, there are code constructs that can be exploited. For instance, if the *snprintf* function from the C standard library is used to process user input without restrictions on the input character set; this function can be used to read from and write to arbitrary memory addresses [Newsham2000]. An analysis of the syntax and the appearance of tokens like *snprintf* can be used as an indicator of whether a function can be exploited on the code level.

The following sections discuss the research fields related to survivability.

## 2.2 Security

Security and Survivability are often mentioned together in the context of *survivability architectures*. But security is only one technique to protect system components [Yurcik2000]. It focuses on how to prevent or detect threats. These are mechanisms that are introduced to a system to put a system into or maintain a system in a state of inviolability from hostile acts or influences. Survivability seen as a system attribute does not introduce any changes to a system. It merely observes the system-state.

Topics that are addressed under the term security include confidentiality, integrity and availability [Avizienis2001]. These are same attributes that usually are targeted by an intelligent adversary. The attack by this intelligent adversary is among the threats a system should be able to survive.

Both concepts, survivability and security, are focused on the concern of protection from malicious intents. Security addresses protection improvement. Survivability, on the other hand, addresses how any kind of change within a system and all its components will affect the systems ability to deliver a certain service. Intuitively, security deals with prevention or recognition of unauthorized behavior, while survivability research deals with behavior issues once a fault has occurred.

## 2.3 Reliability

Reliability  $R(t)$ , is defined as the conditional probability that a component operates correctly throughout the interval  $[t(0), t(n)]$  given that it was operating correctly at time  $t(0)$  [Johnson1989]. Reliability is a function of the number and types of faults that can occur. Looking at the classification of faults, the following assumptions are generally made: faults occur independently as well as randomly. These assumptions draw a distinction between survivability and reliability. Faults can be divided into three categories: permanent faults, intermittent faults, and transient faults [Johnson1989]. Permanent faults are the most likely ones to occur, followed by intermittent faults. The least likely category of faults is transient faults. These faults do not follow a certain pattern, they might occur once and never again afterwards. This order of likelihood according to fault type only holds if independence and randomness is assumed. If an intelligent source is coordinating these faults, the order does not hold.

Reliability can only be stated correctly in the context of the operational profile. An example is given in [Rivers1998]:

...probability that a real-time system will give specified functional and timing performance for the duration of a ten hour mission when used in the way and for the purpose intended. The usage information is quantified through operational profiles.

Survivability research focuses on the fact that software usage can happen outside the operational profile.

## 2.4 Fault and Intrusion Tolerance

As the word *tolerance* implies, the goal of the research in these fields is to find mechanisms that allow a system to tolerate a known threat or fault. The assumption is that not every fault can be avoided, but with enough information about their nature, the fault will not result into a system failure.

Fault tolerant techniques include the detection of an error, damage assessment, reconfiguration, and eventually recovery. Intrusion tolerance is based on fault tolerance and intrusion detection [Cheung1999]. Fault tolerance as well as intrusion tolerance include methods and techniques that are applied to improve an existing solution to increase system attributes like reliability or availability. For a given system, this can result in an increase in survivability, as well.

Given the definition of reliability, this means that the probability that a system will function in a fixed time interval can be increased by invoking fault tolerant approaches. One outcome of the research introduced in this paper will be to show that the same relationship must exist for survivability and intrusion tolerance. Intrusion tolerance and fault tolerance research are somewhat overlapping. In fault tolerance, faults are generally assumed to be independent and their activation unintentional. This assumption of independence and unintentionality is not made for intrusion tolerance, but the fault types that can lead to a failure are the same.

## 2.5 Fault and Intrusion Avoidance

The basic idea of fault avoidance is to prevent faults due to design and implementation mistakes from occurring in the first place. Fault tolerance, high availability or reliability can not be ensured by only measuring and testing, but have to be designed into the system. This implies that there has to be an a priori awareness about which faults could happen and how these can avoided in the first place. Selecting high quality components, enforcing design rules, and reviewing and verifying the result of development steps are examples of fault avoidance techniques [Johnson1989]. While fault tolerance deals with faults once they occur, fault avoidance is used to prevent faults from being designed into the system [Johnson1989].

Intrusion avoidance discovers techniques to prevent attacks from happening. It can be seen a sub-discipline of security, and includes areas like authentication, access control and encryption. A survivability measure can be a tool to aid the design of fault and intrusion avoidance methods. It can support the decision making process about whether a goal was reached in terms of avoidance or if improvements are required by measuring the survivability before and after any changes to the system.

## 3 Measuring Code-Level Survivability

Survivability is an attribute. For each distinct observed entity there can be a value assigned to this attribute. The question is how can this value be found. Survivability is an external metric in terms of measurement. It can only be approximated through internal metrics.

This paper discusses the framework for a metric that will allow this approximation for the survivability of software. The smallest unit that is measured is the module, as introduced in

section 2.1. The metric should relate certain constructs and tokens within the source code of a module to the survivability attribute. Therefore, internal attributes have to be found which influence the survivability of software. Changes to those attributes will change the survivability of the system on the source code level. The existence, absence, and values of these internal attributes can be used to state the degree of survivability.

### 3.1 Software Reliability Measurement

Software reliability engineering is the applied science of predicting, measuring, and managing the reliability of software-based systems to maximize customer satisfaction [Musa1990]. Some areas of software reliability engineering include predicting reliability from characteristics from both the product and the development process or estimating it from failure data in test, based on models and expected use [Musa1990].

Ultimately, the reliability of a product is only known after the product has gone out of service. At this point it is known when and hopefully why the software failed; how reliable it was. This can only be used to validate predictions about product quality, but not to estimate the quality beforehand.

Software reliability can be expressed as the mean time to failure (MTTF) or mean time between failures (MTBF). There are generally two ways to approximate these numbers. MTTF and MTBF can be estimated through testing according to the operational profile. Furthermore, the MTTF can be approximated through measurement of internal attributes of the code and/or other parts of the software creation process.

The second methodology allows the creation of a relative measure of reliability before the actual MTTF data is available. For example, it allows deciding which of a set of alternative solutions to the same problem should be included in the final software product or to support decision making about redesign. The technique attempts to identify internal attributes of software that influence the reliability of the final product [Hanebutte2000]. A relative measure of an external attribute does not give the exact MTTF or reliability, but it allows ordering entities according to their attribute value. Furthermore, it shows if a redesign increased (or decreased) the count for this measure. This allows one to state if any changes are actually improving the entity in terms of the observed attribute.

### 3.2 The Attributes

The idea of a survivability metric is based on the fact that certain code constructs make source code more vulnerable to malicious input. These code constructs, which can be single keywords as well as fixed sequences, can be counted. Previous research and documentation has shown that most of the code exploited has very similar features that made the code particular vulnerable [CERT].

This research will focus on potential vulnerable functions within C. For example, the majority of buffer overflows are due to the usage of C functions like *memcpy* and *strcpy*. These are functions that don't perform boundary checking when copying values into main memory and therefore can overwrite parts of the memory and alter the behavior of the application in a way that it is outside the operational profile. Furthermore, there are functions from the *printf*

family that, when confronted with special input characters, allow access to arbitrary memory addresses.

Theoretically, anytime a value is accessed by an application, a function could be confronted with malicious data. Input can be performed through, but is not limited to:

- user input
- file access
- access to shared memory
- incoming network traffic

Additionally, there is a problem with temporary data, which is created as part of the execution method but not intended to be observed by a third party, including:

- temporary files
- network traffic
- temporary variables

Finally, a program can be interacting with other software or a privileged user. Its outgoing traffic could be intercepted for information retrieval or even to modify the transferred information. This can be:

- outgoing network traffic
- output to configuration files
- output to shared memory

The scope of this work includes neither malicious alteration of source code outside the specified requirements nor the incorrect implementation of requirements. It is assumed that the observed software is not altered in terms of changes to the operational profile.

It can be said that survivability has to be measured especially around the interfaces of an application. Attributes to be counted are the usage of C library functions, which do not perform thorough length checking. These set of functions include *sprintf()*, *memcpy()*, *strcpy()* and, *gets()*. Other functions, such as members of the *printf* family, are rather sloppy in the way they process input and can therefore be used to access arbitrary memory.

The usage of the *system()* and members of the *exec()* family are other survivability attributes. If these are used within the source code of a program that is run as *suid* root, the invoked program will be run as the root user as well. This invoked program can then hold the exploitable vulnerability.

A program that inputs and outputs data many times is more likely to have an access point for a malicious intent. Here the number of input and output operations can be counted.

Furthermore, the complexity of software should be taken into consideration. Complex and long source code modules are more likely to contain faults, since they are usually more difficult to test and to audit for faults.

It cannot be said that if any of the constructs described above are part of a piece of code, this code is automatically more vulnerable. However, it is valid to assume that the likelihood of exploitability of code increases with an increased count of these measures.

Therefore, a set of trade-off measures is necessary to weigh the found tokens. For instance, in [Rosenberg1997] the metric Source Lines of Code (SLOC) is suggested as a covariate to “normalize” other measures. The covariate adjusts the measures of the number of potential vulnerable tokens found for the size, complexity, and information flow of the measured module. For instance, if a number of potentially vulnerable function calls were found in a simple module, the chances that these are securely used are higher than the same number found in a highly complex or cryptically written program.

Measures of code size, complexity and module communication will therefore be used as covariates. A set of 14 quality measures that can be used are discussed in [Hanebutte2000]. These measures include derivatives of McCabe's Cyclomatic complexity [McCabe1976], Halstead's number of unique operators and operands [Halstead1977], and Zage's control flow metrics [Zage1993].

All metrics should be combined into a survivability index. The index will consist of a set of measures that state the number of potentially vulnerable function calls found. The impact of these numbers are weighted or “normalized” by a set of covariate metrics from the domains: size, complexity and module communication.

The relationship between all these metrics is analyzed and correlation is to be removed to allow the metrics to be input into a multi-regression model (Figure 1).

The survivability index is merely a relative rather than an absolute measure. It can put source code modules into a partial order in terms of survivability. This allows comparison of alternative code solutions for the same task. Theoretically, thresholds can be assigned as knowledge and understanding of the behavior of a module emerges, e.g. an experiment shows how many real vulnerable code sequences are in the source code and which index was measured for the same code.

The raw metrics need to be combined into the survivability index. This usually requires that the measures are screened using Principle Component Analysis and if a high correlation among the raw metrics exists, Factor Analysis can be used to build abstract metric groups. These independent groups can be combined into a multi-regression model. Factor Analysis is required to ensure that regression model assumption of variable independence is not violated.

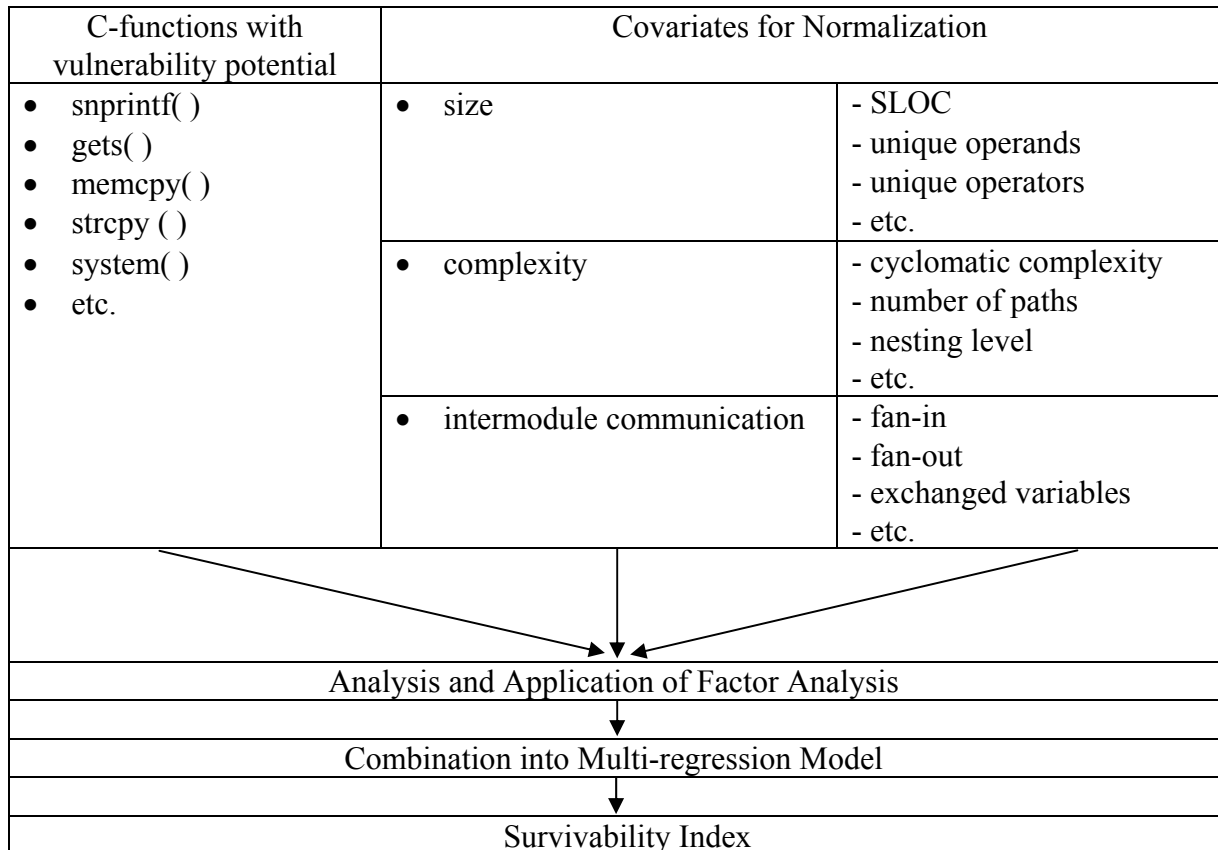
#### 4 Future Perspective – a Real-time Survivability Measure

The Survivability Lab at the University of Idaho created a technique to instrument the Linux kernel or any other application. The instrumentation inserts so-called “hooks” into the code. This allows access to the number of calls to any instrumented function within the source code for any desired unit of time during runtime. If the survivability index for an instrumented module is known, it is possible to state an overall dynamic system survivability  $\xi_{sys}$  by combining the frequency of a module with its survivability-index  $I_n$ .

For a given system at the end point of each time interval  $t$ , the system state can be stated as a function  $f$  of all modules frequencies  $\phi_n$  and the survivability index for each module. The module frequency is number of times a module was invoked within a measured time interval.

$$\xi_{\text{sys}}(t) = f((\phi_1, I_1), (\phi_2, I_2), \dots, (\phi_n, I_n))$$

where  $n$  is number of instrumented modules.



**Figure 1:** Sample Metrics to be combined into a Survivability Index

This would allow observation of the availability-like measure – the *dynamic survivability*.

Therefore, for each time interval, it would be known how vulnerable the system is in the context of usage, environment and ongoing threats. For example, if several instances of a module with a low survivability are invoked, additional security measures could be taken. Another option could be that the system state is not allowed to fall below a certain  $\xi_{\text{sys}}$ , and if it does fall below this threshold certain non-essential services will not be available until the overall  $\xi_{\text{sys}}$  rises again above a defined threshold.

## 5 Summary

This paper describes an attempt to analyze the meaning of term survivability with the goal of quantification of system survivability. Survivability is a newer research field, which has gotten increased attention in the last decade in computer science, when it was realized that



absolute system security is unattainable. The goal of survivability research is to create systems that are capable of providing essential and reliable services even though parts of the system are down or compromised.

Quantification or measurement survivability would allow the verification if an attempt to increase the system survivability was successful or if a change to the system had an impact to its survivability.

This paper discusses a survivability measure at the source code level. The source code module is identified as the smallest measurable unit. A system is therefore composed out of these units. Therefore, a system's survivability should be stated as a composite measure from the measures for each unit used. The metrics that build the survivability index are those tokens that have been identified in past research [CERT] to be likely to yield vulnerabilities. Furthermore, measures of module complexity, size, and coupling should be part of a survivability metric as covariates.

## References

- [Aviziens2001] A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability, Research Report N01145, LAAS-CNRS, April 2001.
- [CERT] CERT/CC Vulnerability Notes Database. <http://www.kb.cert.org/vuls/>
- [Cheung1999] Steven Cheung, *An Intrusion Tolerance Approach for Protecting Network Infrastructures*. PhD thesis, UC Davis 1999.
- [Dietrich2002] S. Dietrich and P. A. Y. Ryan. The survivability of survivability. In *Proceedings of the 4<sup>th</sup> Information Survivability Workshop, (ISW-2001/2002)*, March 18-20 2002.
- [Elbaum1999] Sebastian Elbaum and John Munson. Intrusion detection through dynamic software measurement. In *Proceedings of the 8<sup>th</sup> USENIX Security Symposium*, August 20-26 1999.
- [Hanebutte2000] Nadine Hanebutte and Reiner R. Dumke. Analyzing software design using a measurable program design language. *Metric News*, 5:23-32, 2000.
- [Halstead1977] Maurice H. Halstead. Elements of software science. *Operating, and Programming Systems Series. 7*. New York, NY: Elsevier, 1977.
- [Johnson1989] Barry W. Johnson. *The Design and Analysis of Fault Tolerant Digital Systems*. Addison Wesley, 1989.
- [Longstaff2002] Thomas A. Longstaff. Survivability as quality. In *Proceedings of the 4<sup>th</sup> Information Survivability Workshop, (ISW-2001/2002)*, March 18-20 2002.
- [McCabe1976] Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering* 4: 308-320, 1976.



- [Mead2000] Nancy R. Mead, Robert J. Ellison, Richard C. Linger, Thomas Longstaff, and John McHugh. Survivable network analysis method. Technical Report CMU/SEI-2000-TR-013 ESC-2000-TR-013, CMU/SEI, 2000.
- [Musa1990] John D. Musa and William W. Everett. Software-reliability engineering: technology for the 1990s. *IEEE Software*, 7:36-43, 1990.
- [Neumann2000] Peter G. Neumann. Practical Architectures for Survivable Systems and Networks. Technical report, SRI, 2000.
- [Newsham2000] Tim Newsham. Format string attacks. White paper, September 2000.
- [Rivers1998] Anthony T. Rivers. *Modeling Software Reliability During Non-Operational Testing*. PhD Thesis, North Carolina State University, 1998.
- [Rosenberg1997] Jarrett Rosenberg. Some misconceptions about lines of code. In *Proceedings of the 4<sup>th</sup> International Software Metrics Symposium, (METRICS'97)*, November 5-7, 1997.
- [US1037] U.S. Department of Commerce. 1037C. In *Federal Standard*. National Telecommunications and Information Administration, Institute for Telecommunications Services, 1996.
- [Voas2002] Jeffrey M. Voas. Ten challenges in information survivability. In *Proceedings of the 4th Information Survivability Workshop, (ISW-2001/2002)*, March 18-20 2002.
- [Yurcik2000] William Yurcik and David Doss, Hans Kruse. Survivability-over-security: providing whole system assurance. In *Proceedings of the 3rd Information Survivability Workshop, (ISW-2000)*, October 24-26 2000.
- [Zage1997] W.M. Zage and D.M. Zage. Evaluating design metrics on large-scale software. *IEEE Software* 7:75-81, 1997.

**Dumke, R.; Rombach, D. (Eds.):**

*Software-Messung und -Bewertung*

*Deutscher Universitätsverlag (DUV), Wiesbaden, 2002, (254 pages)*

*ISBN 3-8244-7592-8*

**This book includes the proceedings of the annual Workshop of the GI FG 2.1.10 held in Kaiserslautern in September 2001. Main categories of the papers collection are**

- National initiatives for a virtual competence center of software engineering knowledge services and web-based experimentation,
- Overviews and new results of the methodologies of functional size measurement and cost estimation,
- New measurement approaches for distributed CORBA-based software, agent-based and object-oriented systems,
- New intentions of software measurement for special aspects such as quality model-based risk analysis, performance engineering and cost estimation,
- Further measurement application for software maintenance and project controlling.

The book will be of interest to software engineering researchers, as well as to practitioners in the areas of project management and quality improvement programs, for both software maintenance and software development in general.

**Schmietendorf, A.; Dumke, R.; Hopfer, R.; Scholz, A. (Eds.):**

*Tagungsband vom 3. Workshop Performance Engineering*

***in der Softwareentwicklung (PE 2002), Mai 2002, Dresden, Germany***

**This proceedings includes the papers from the annual Workshop on Performance Engineering (PE2002) held in Dresden in May 2002. The main topics are**

- Software agent-based performance tuning and controlling,
- Performance analysis of enterprise application integration (EAI) solutions,
- Software systems speed management,
- Performance engineering of agent-based systems and system development.

This proceedings can order by a message to Reiner Dumke, University of Magdeburg, Faculty of Informatics.

**Dumke, R.; Bundschuh, M. (Eds.):**

*Software-Metriken in der Praxis*

Shaker Verlag, Aachen, 2002, (154 pages)

ISBN 3-8322-0470-9

This book includes the papers or slices from the presentations of the annual DASMA conference as **Metrikon 2001** held in Dortmund in October 2001. Some of the conference topics are

- Software metrics and project controlling as general approach and as lesson learned from long time experience,
- Psychological aspects of software metrics applications,
- Metrics-based quality analysis of large object-oriented software systems,
- Automation in sourcecode analysis and metrics databasis,
- GQM applications and experiences in industrial environments.

**For more information please contact the DASMA office or see the contents description in the GI FG2.1.10 home page (<http://ivs.cs.uni-magdeburg.de/us/giak/>).**

**Dumke, R.; Rautenstrauch, C.; Schmietendorf, A.; Scholz, A. (Eds.):**

*Performance Engineering. State of the Art and Current Trends*

LNCS 2047, Springer-Verlag, Heidelberg 2001 (349 pages)

ISBN 3-540-42145-9

The performance analysis of concrete technologies has already been discussed by a multitude of publications and conferences, but the practical application was often neglected. An engineering procedure was comprehensively discussed for the first time on the “International Workshop on Software and Performance: WOSP 1998” in Santa Fe, NM in 1998. Teams were formed, which examined the integration of performance analysis into the software engineering in particular. Practical experiences from industry and new research approaches were discussed in these teams. Diverse national and international activities, e.g., the foundation of a working group within the German Association of Computer Science followed.

This book continues the discussion of performance engineering methodologies. On the one hand, it is based on selected and revised contributions of conferences that were carried out in 2000:

- Second International Workshop on Software and Performance - WOSP 2000, September 17 – 20 in 2000 in Ottawa, Canada,
- First German Workshop on Performance Engineering within the Software Development May 17<sup>th</sup> in Darmstadt, Germany.

On the other hand, further innovative ideas were considered by a separate call for chapters. With this book we would like to illustrate the state of the art, current discussions and development trends in the area of performance engineering.

In the first section of the book, the relation of software engineering and performance engineering is discussed. In the second section, the use of models, measures and tools is described. Furthermore, case studies with regard to concrete technologies are discussed in the third section.

The contributions published in this book underline the international importance of this field of research. 20 contributions were considered from Venezuela, Spain, Cyprus, Germany, Canada, USA, Finland, Swedes and Austria.

Juristo, N.; Moreno, A.M.:

**Basics of Software Engineering Experimentation**

Kluwer Academic Publishers, 2001 (395 pages)

ISBN 0-7923-7990-X

**Basics of Software Engineering Experimentation is a practical guide to experimentation in a field which has long been underpinned by suppositions, assumptions, speculations and beliefs. It demonstrates to software engineers how Experimental Design and Analysis can be used to validate their beliefs and ideas.**

**The book does not assume its readers have an in-depth knowledge of mathematics, specifying the conceptual essence of the techniques to use in the design and analysis of experiments and keeping the mathematical calculations clear and simple.**

**Basics of Software Engineering Experimentation is practically oriented and is specially written for software engineers, all the examples being based on real and fictitious software engineering experiments.**

**“If you are a researcher, you should master the approaches to empirical software engineering described by Juristo and Moreno ...**

**If you are a practitioner, the advice in this book will enable you to read and assess the studies you find in your journals and at your conferences**

...

**If you are an educator, this book will help you to guide your students in understanding that software engineering is far more than simply having a good technology idea and trying it out on a project.” (Pfleeger, S.L.)**

**PE2002:**

**3<sup>rd</sup> Workshop on Performance Engineering für die Softwareentwicklung**  
May 22, 2002, HfTW Dresden, Germany  
see: <http://www-wi.cs.uni-magdeburg.de/pe2002/>

**Metrics 2002:**

**8<sup>th</sup> International Symposium on Software Metrics**  
June 4 – 7, 2002, Ottawa, Canada  
see: <http://www.software-metrics.org/>

**WOSP 2002:**

**Third International Workshop on Software and Performance**  
July 24 – 26, 2002, Rome, Italy  
see: <http://univaq.it/~wosp02/>

**CONQUEST 2002:**

**Conference on Quality Engineering in Software Technology**  
September 18 – 20, 2002, Nuremberg, Germany  
see: <http://www.asqf.de/deu/conquest/2002/>

**IFPUG 2002:**

**Annunal IFPUG Conference**  
September 24 – 27, 2002, San Antonio, Texas  
see: <http://www.ifpug.org/conferences/call.htm>

**IWSM 2002:**

**12<sup>th</sup> International Workshop on Software Measurement**  
Ocotber 7 – 9, 2002, Magdeburg, Germany  
see: <http://iws2002.cs.uni-magdeburg.de/>

**IWCMQ 2002:**

**International Workshop on Conceptual Modeling Quality**  
October 7-11, 2002 Tampere, Finland  
see: <http://alarcos.inf-cr.uclm.es/iwcmq02/>

**EuroSTAR 2002:**

**10<sup>th</sup> European International Conference on Software Testing Analysis & Review** November 11 - 15, 2002, Edinburgh, Scotland  
see: <http://www.testingconferences.com/eurostar/home/>

**WWW 2003:**

**International World Wide Web Conference**  
May 20-24, 2003, Budapest, Hungary  
see: <http://www2002.org/> with the Alternate Track *Web Engineering*  
<http://webengineering.org/events/>

**ASM 2003:**

Applications of Software Measurement  
June 2-6, 2003, San Jose, CA  
see: <http://www.sqe.com/asm/events.asp>

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.gucis.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/lrg1.html](http://saturne.info.uqam.ca/Labo_Recherche/lrg1.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://www.trese.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>  
Annotated 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](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](mailto:news:comp.software-eng)
- [news:comp.software.testing](mailto:news:comp.software.testing)
- [news:comp.software.measurement](mailto:news:comp.software.measurement)

### **Software Measurement Associations**

- <http://www.aemes.fi.upm.es>  
AEMES Association Espanola de Metricas del Software
- <http://www.asqf.de>  
ASQF Arbeitskreis Software-Qualität Franken e.V., Nuremberg, Germany
- <http://www.cosmicon.com>  
COSMIC Common Software Measurement International Consortium
- DANMET: Danish Software Metrics Association
- <http://www.dasma.de>  
DASMA Deutsche Anwendergruppe für Software Metrik und Aufwands-schätzung e.V.
- <http://www.esi.es>  
ESI European Software Engineering Institute in Bilbao, Spain
- <http://www.fesma.org/>  
FESMA Federation of European Software Metrics Associations
- <http://www.sttf.fi>  
FiSMA Finnish Software Metrics Association

- FFPUG: French Function Point User Group
- FPUGA: Function Point User Group Austria
- <http://www.iese.fhg.de>  
IESE Fraunhofer Einrichtung für Experimentelles Software Engineering
- <http://www.isbsg.org.au>  
ISBSG International Software Benchmarking Standards Group, Australia
- <http://www.nesma.nl>  
NESMA Netherlands Software Metrics Association
- <http://www.sei.cmu.edu/>  
SEI Software Engineering Institute Pittsburgh
- <http://www.spr.com/>  
SPR Software Productivity Research by Capers Jones
- <http://fdd.gsfc.nasa.gov/seltext.html>  
SEL Software Engineering Laboratory - NASA-Homepage
- <http://www.vrz.net/stev>  
STEV Vereinigung für Software-Qualitätsmanagement Österreichs
- <http://www.sqs.de>  
SQS Gesellschaft für Software-Qualitätssicherung, Germany
- <http://www.ti.kviv.be>  
TI/KVIV Belgisch Genootschap voor Software Metrics
- <http://www.ukσμα.co.uk>  
UKSMA United Kingdom Software Metrics Association

### **Software Metrics Tools (Overviews and Vendors)**

#### **Tool Listings**

- <http://www.pitt.edu/~ddarcy/isprof/intotool.html#intro>  
Metrics Tool Listings by Dace Darcy
- <http://www.cs.umd.edu/users/cml/resources/cmetrics/C/C++ Metrics Tools by Christopher Lott>

- <http://davidfrico.com/mettools.htm>  
Software Metrics Tools by Dave
- <http://mdmetric.com/meast11.htm>  
Maryland Metrics Tools
- <http://cutter.com/itgroup/reports/function.html>  
Function Point Tools by Carol Dekkers

### **Tool Vendors**

- <http://www.mccabe.com>  
McCabe & Associates
- <http://www.scitools.com>  
Scientific Toolworks, Inc.
- <http://zing.ncsl.nist.gov/webmet/>  
Web Metrics
- <http://www.globalintegrity.com/csheets/metself.html>  
Global Integrity
- <http://www.spr.com/>  
Software Productivity Research (SPR)
- <http://jmetric.it.swin.edu.au/products/jmetric/>  
JMetric
- <http://www.imagix.com/products/metrics.html>  
Imagix Power Software
- <http://www.verilogusa.com/home.htm>  
VERILOG (LOGISCOPE)
- <http://www.qsm.com/>  
QSM

# METRICS NEWS

---

VOLUME 7

2002

NUMBER 1

---

## CONTENTS

<b>Call for Participation .....</b>	<b>3</b>
<b>Position Papers .....</b>	<b>7</b>
<i><b>Manfred Bundschuh:</b></i> Function Point Prognosis Approved – Regression Analysis for Approximation of Function Point Counts .....	<b>7</b>
<i><b>Nadine Hanebutte:</b></i> Quantification of Survivability at the Code Level Through Software Measurement .....	<b>15</b>
<b>New Books on Software Metrics .....</b>	<b>27</b>
<b>Conferences Addressing Metrics Issues .....</b>	<b>31</b>
<b>Metrics in the World-Wide Web .....</b>	<b>33</b>

---

**ISSN 1431-8008**