[Fel96] T. Fellger. *Structuring and classification of experience knowledge with hypertext-based realization*. Master's thesis. University of Mannheim, June 1996 (in German)

[GHJ+94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of reusable object-oriented software*. Addison-Wesley, Reading, Massachusetts, 1994

[GR96] C. Differding and H.D. Rombach. *Continual software quality improvement in industrial practice*. In C. Ebert and R. Dumke (Eds.), Software metrics in practice, Springer-Verlag, Berlin, 1996, pp. 14-44 (in German)

[Kru92] C.W. Krueger. *Software reuse*. ACM Computing Surveys, 24, 1992, pp. 131-183

[Min87] B. Minto. *The pyramid principle - logic in writing and thinking*. Minto International Inc., London, 3rd rev. edition, 1987

[PEF96] *The Perfect Experience Factory Model*. A booklet form the PERFECT ESPRIT project 9090 handbook edition, 1996

[Pri91] R. Prieto-Díaz. *Implementing faceted classification for software reuse*. Communications of the ACM, 34 (5), 1991, pp. 88-97

[SEL95] *Software process improvement guidebook*. Software Engineering Laboratory Series, SEL-95-102, March 1996. On-line available at http://fdd.gsfc.nasa.gov/selspig.pdf.

Validation Process in Software Engineering: An example with Function Point

Jean-Marc Desharnais, Pam Morris

1. Background

Benchmarking[1] studies based on actual data supplied by many industrial sites require two types of validation: validation of the collection process as well as validation of the data supplied to the researchers. The former validation will be referred to as *a priori* validation and the later one as *a posteriori* validation. The *a priori* validation will review all the steps and procedures of the data collection process within benchmarking data. The *a posteriori* validation will establish the degree of quality of the data collected and, wherever required, will eliminate from the comparative studies the data deemed not reliable. Both types of validation are required to ensure the quality and the integrity of the data for meaningful benchmarking studies in software engineering due the current immaturity of both software metrics and software engineering processes at industrial sites as assessed the maturity model developed by the Software Engineering Institute (USA).

2. A Priori Validation

The *a priori* validation must look into the data collection process of each site supplying data for the benchmarking studies. It should include the following steps:

1. Verification, for each site, of the existence of written, accurate and reliable documentation on metrics definitions, coding schemes and admissible values.
2. Verification of consistency between the written documentation and current data collection practices of the sites.
3. Verification of the existence of training and /or coaching programs to ensure that the staff collecting the metrics have a solid understanding of software metrics that they collect.
4. Verification that responsibilities are clearly defined for data coding, data gathering and data storage process.
5. Verification of the existence and effectiveness of quality controls.

[1] Desharnais, Jean-Marc, Validation Process for Industry Benchmarking Data, Software Engineering Laboratory in Applied Metrics, Invited Paper, IEEE Conference on Software Maintenance, sept.-oct 1993, Montréal.

Whenever possible, there should be also steps included to identify if there exist other sources of information against which some of the information can be cross-validated.

3. A Posteriori Validation

The *a posteriori* validation should include an analysis of the data sets themselves. It should include the following three steps:

Step 1: Relevant data
Verify that the data sets supplied by the industrial sites includes all the required data variables. Furthermore, the data sets must be scrutinized for irrelevant data, which must then be discarded.

Step 2: Data definition consistency
Verification that the data definitions and coding schemes are consistent across sites and over time. If not, corrective action should be taken, including partial grouping of data if deemed appropriate.

Step 3: Validity of data valued within the data sets
This last step includes many sub-activities:

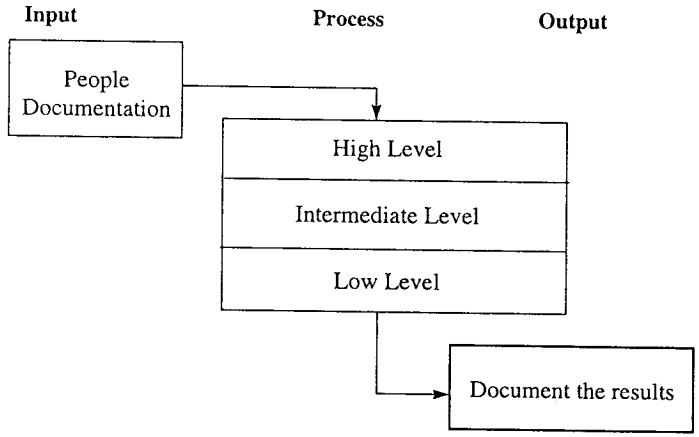
a) Definition of validation criteria for each variable (eg. expected minimum and maximum);
b) Identification of outliers;
c) Validation of outliers with site staff.

In addition to the above two types of validation, site staff feedback should be obtained on the accuracy and reliability of the data and data collection processes. This could lead to the identification of biases within the data supplied for the benchmarking studies.

4. Post Measurements Validation Procedure for the Function Point Counts

The post measurements validation procedure for Function Point counts is a "a posteriori validation". Although Function Point Analysis has been in existence for almost 20 years, the idea of having a systematic procedure to validate the resultant Function Point count "a posteriori" has just started to arise. A presentation on this topic was delivered at the IFPUG 1996 spring Conference in Atlanta[2] and an article relating to the same subject in the first issue of the Voice, a Publication of the International Function Point Users Group[3] in June 1996. Similar ideas were also expressed in the 1994 Ph.D. thesis of Alain Abran[4].

The A posteriori validation process covers the following areas:



4.1 Preparation

- prepare for the validation process by identifying the major participants in the count (e.g. Function Point Specialist and Subject Matter Expert),
- ensure that a complete set of information is available to the review (e.g. data model and process model),
- ensure that a complete set of procedures is available to the review (e.g. a clear set of official rules and "local rules[5]" to count Function Points).

[2] Morris Pam, Desharnais Jean-Marc, Validation of Function Point: an experimental perspective, IFPUG Conference, April 1996.
[3] Longstreet David, Auditing Function Point Counts, The Voice, A Publication of the International Function Point Users Group, 1996, pp. 8-9-10.
[4] Abran, Alain, "Analyse du processus de mesure des points de fonction", Ph.D. Thesis, École Polytechnique de Montréal, Mars 1994, 405 pages.
[5] The word local rules refer to the adaptation of official rules to the documentation available in a specific site or for a specific application/project.

4.2 High level review

The high level review is aimed at identifying any major strategy errors employed in the count. The *high* level review examines the :

- ◊ unadjusted functional size of the software being measured,
- ◊ scope of the Function Point count [6],
- ◊ boundary to the software being measured.

4.3 Intermediate level review

This level of the review explores these intrinsic relationships between inputs, files and outputs/Inquiries for the measured software and compares them with values collected from industry.

More specifically it examines the :

- ◊ relationship between the number of logical files and the unadjusted functional size of the software.

Figure 1 shows the relation between the number of data files (Internal Logical Files + External Interface Files) and the total number of Function Points. Typically data (ILFs and EIFs)) contribute about 30% of the total number of function points, whilst transactions (input, output and inquiry) contribute to the remaining 70%.

[6] For a development project and a major enhancement project the scope is the functionality impacted by the project activities. The functionality is confined by the limits of the software's external application boundary (refer ISO/IEC/SC7 Standard DIS 14143 "Software Measurement - Definition of Functional Size").

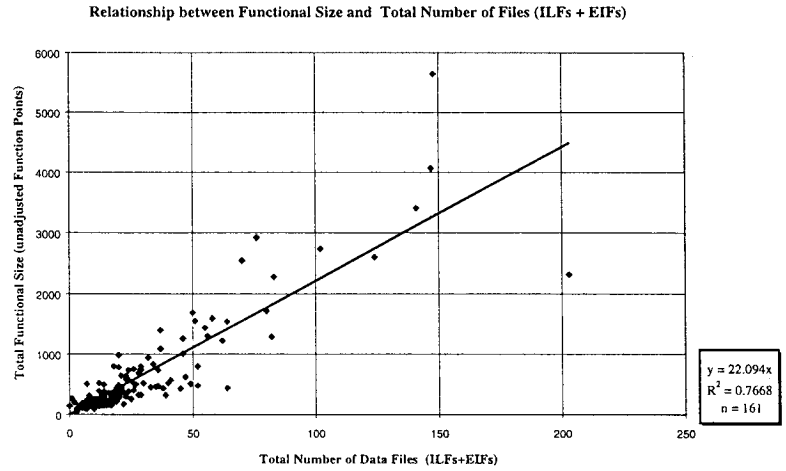


Fig. 1

- ◊ relationship between the numbers of functions to each other,

Figure 2 shows the relationship between Inputs, Outputs and Inquiries from the authors' database (a combination of Australian and Canadian data).

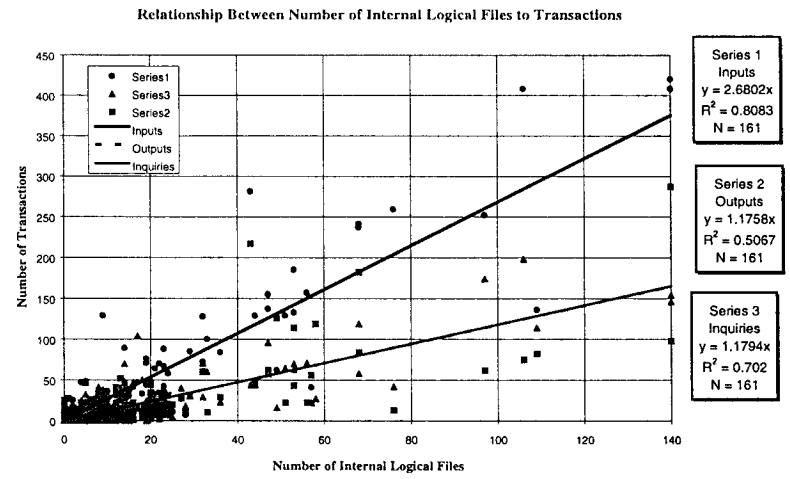


Fig. 2

◊ percentage contribution of each function type to the unadjusted functional size of the software,

Figure 3 shows the percentage contribution to total Size by Function Type from ISBSGs[7] an Australian database (external ring) and the authors' database (internal ring).

Percentage Contribution to Total Size by Function

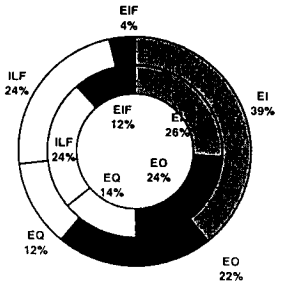


Fig. 3

◊ profile of the complexity of the functions.

This check also explores the intrinsic nature of MIS applications. The following table compares the mean function points awarded for each function type found within the authors' database and the ISBSGs database with the equivalent complexity rating in the IFPUG complexity matrices. In both databases the overall mean complexity for :

◊ Transactional function types is closest to 'average',

◊ Data function types is closest to 'low'.

[7] International Software Benchmarking Standard Group Repository Release 2 October 1995.

Function Type	Mean Function Points Awarded (Total number FPs /Total number functions)		Corresponding IFPUG Complexity Rating	
	Dataset 1 (n=106)	Dataset 2 (n = 161)	Average	Low
Inputs	4.3	4.2	**4**	***3***
Outputs	5.4.	5.8	**5**	***4***
Inquiries	3.8	4.0	**4**	***3***
Internal Logical Files	7.4	7.8	***10***	**7**
External Interface Files	5.5	5.2	***7***	**5**

Tab. 1

4.4 Low level review

The *lowest* level review examines the :

◊ function point count of selected Data Business Functions,

◊ function point count of selected Transaction Business Functions.

This level of the review is performed when the previous level reviews have assessed the count as being potentially :

valid - then the detailed review is used to confirm the assessment, or

invalid - then the detailed review is used to investigate sources of error to be corrected.

For example:

If the number of function points contributed by ***External Inputs*** is outside the normal expected ranges and is:

- Too **high,** then check for the following common counting errors or reasons for variances:
  - ◊ input selection filters for reporting data may have been incorrectly counted as External Inputs,
  - ◊ physical data entry screens may have been incorrectly counted as logical transactions (elementary processes),
  - ◊ each slight variation in the data entered for an input transaction may have been incorrectly counted as a unique elementary process,
  - ◊ physical transactions eg. save, exit, send may have been incorrectly counted as logical External Input transactions.
- Too **low,** then check for the following common counting errors or reasons for variances:

- ◊ add, change and delete functions may have been bundled into a single maintenance input and not counted as discrete elementary processes,
- ◊ counter may have not looked for additional input transactions beyond standard maintenance eg. *Cancel Invoice , Post Invoice to GL* etc.,
- ◊ an incoming file has been incorrectly counted as a single External Input instead of evaluating each transaction type for qualification as a unique elementary process,
- ◊ physical files have been incorrectly grouped into logical files causing both the complexity of the files to be incorrect and the complexity of every transaction accessing the logical file to be incorrect,
- ◊ business rules may not allow the user to delete data.

4.5 Documentation of the Validation Process

The main areas addressed are:

- ◊ review the detail of the documentation and ensure that is has followed prescribed documentation standards,
- ◊ verify if the documentation supplied with the count is detailed enough to allow:
  - all functions which contributed to the count to be identified,
  - all assumptions and decisions made during the count to be verified,
  - another Certified[8] function point counter to easily update the count.

An example of the collection and codification standards for count documentation as follows:

Count Data collection procedures:

Each document produced by the count must include the relevant page number on the lower right corner of every page.

Count codification Procedures:

The count will be recorded using the level of detail and the documentation coding conventions described below:

[8] A Certified counter is a person who is accredited by IFPUG as a Certified Function Point Specialist.

Function type:

Each function type is identified by a letter

I: External Input

O: External Output

Q: External Inquiry

ILF: Internal Logical File

EIF[9]: External Interface File

Each function type code will have a suffix. This suffix will identify the order of the function type on a page, for example, if the first function type identified on a page is an EI:"I1" will be written next to the EI. If the second function type identified on that same page is an EQ: "Q2" will written next to the EQ.

- ◊ the comments made by the persons who performed the count,

Verify that any areas within the count which could be open to interpretation and would affect the count result have been adequately documented.

5. Conclusion

We have proposed and illustrated a standard process for the validation of Function Point results. We put the emphasis on the a posteriori validation which has three steps: the relevance of the data, the consistency of data definitions and coding schemes and the validity of data values within the data sets. This validation process will significantly improve the quality of the data collected.

[9] These acronyms correspond to the IFPUG standard.

GABLER EDITION WISSENSCHAFT
Information Engineering und
IV-Controlling
Herausgegeben von Professor Dr. Franz Lehner

Die Schriftenreihe präsentiert aktuelle Forschungsergebnisse der Wirtschaftsinformatik sowie interdisziplinäre Ansätze aus Informatik und Betriebswirtschaftslehre. Ein zentrales Anliegen ist dabei die Pflege der Verbindung zwischen Theorie und Praxis durch eine anwendungsorientierte Darstellung sowie durch die Aktualität der Beiträge. Mit der inhaltlichen Orientierung an Fragen des Information Engineerings und des IV-Controllings soll insbesondere ein Beitrag zur theoretischen Fundierung und Weiterentwicklung eines wichtigen Teilbereichs der Wirtschaftsinformatik geleistet werden.

Franz Lehner/Reiner Dumke/
Alain Abran (Eds.)

Software Metrics

Research and Practice in Software Measurement

DeutscherUniversitätsVerlag

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Software Metrics : research and practice in software Measurement
/ Hrsg.: Franz Lehner ... .
- Wiesbaden : Dt. Univ.-Verl. ; Wiesbaden : Gabler, 1997
(Gabler Edition Wissenschaft : Information Engineering und IV-Controlling)
ISBN 3-8244-6518-3

Der Deutsche Universitäts-Verlag und der Gabler Verlag sind Unternehmen der Bertelsmann Fachinformation.

Gabler Verlag, Deutscher Universitäts-Verlag, Wiesbaden

Lektorat: Claudia Splittgerber

http://www.gabler-online.de

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt.

Druck und Buchbinder: Strauss Offsetdruck, Mörlenbach
Printed in Germany

ISBN 3-8244-6518-3

Foreword

There ubiquity of software in the development of complex systems and high technology consumer products is steadily increasing in a competitive environment that is now global in scope. Thus software developers are faced with the challenge of making software systems and products of ever greater quality and safety, while at the same time being faced with the growing pressure of costs reduction in order to gain and maintain competitive advantages. As in any scientific and engineering discipline, reliable measurement is essential for talking on such a challenge.

"Software measurement is an excellent abstraction mechanism for learning what works and what doesn't" (Victor Basili). Measurement of both software processes and products provides a large amount of basic information for the evaluation of the software development processes or the software products themselves. Examples of recent successes in software measurement span multiple areas, such as evaluation of new development methods and paradigms, quality and management improvement programs, tool-supporting initiatives and company-wide measurement programs.

The German Computer Science Interest (GI) Group of Software Metrics, the (GI) Interest Group of Quality Improvement of Object-Oriented Systems and the Canadian Interest Group in Software Metrics (CIM) have attended to these concerns in the recent years. Research initiatives were directed initially to the definition of software metrics and then to validation of the software metrics themselves. This was followed by more and more investigation into practical applications of software metrics and by critical analysis of the benefits and weaknesses of software measurement programs. Key findings in this area of software engineering have been published in some important books, such as Zuse's *Software Complexity*, Állerburg and Abran's *Metrics in Software Evolution*, Dumke and Zuse's *Theory and Practice of Software Measurement*, Dumke et al.'s *Software Quality with Metric Tools*, Lehner's *Software Maintenance* and Ebert and Dumke's *Software Metrics in Practice*.

This new book includes key papers presented at the 6th International Workshop on Software Metrics in Regensburg (Germany), September 1996. It is a collection of theoretical studies in the field of software measurement as well as experience reports on the application of software metrics in Canadian, Austrian, Belgian and German companies and universities. Some of these papers and reports describe new software measurement applications and paradigms for knowledge-based techniques, maintenance service evaluation, factor analysis discussions and neural-fuzzy applications. Others address the object-oriented paradigm and discuss the application of the Function Point approach to an object-oriented design method, the evaluation of the Java development environment, the analysis of quality and productivity improvements of object-oriented systems, as well as the definition of the metrics of class libraries. Other papers offer a different perspective, presenting a software measurement education system designed to help improve the lack of training in this field, for example, or they include experience reports about the implemention of measurement programs in industrial environments.

This 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.

Alain Abran
Reiner Dumke
Franz Lehner

Table of Contents

I. Quality and Measurement of Object-Oriented Software