Analyzing Empirical Data from a Reverse Engineering Project

P. Fiore(*), F. Lanubile(**) and G. Visaggio(**)

(*) Basica spa Via G. Amendola 162/1 70126 Bari, Italy
Tel. 39-80-5484555 Fax. 39-80-5484556
(**) University of Bari, Italy Dipartimento di Informatica
Via Orabona, 4 70126 Bari, Italy
e-mail: visaggio@seldi.uniba.it

Abstract[1]

This study reports the analysis of data collected during the execution of a reverse engineering process in a real environment. Essentially, the analysis assessed productivity aspects. The experience showed the need for automatic tools which can be adapted to the context the process is performed in and the possibility of anchoring the effort required for the reverse engineering to the desired product quality.

1. Introduction

Although there have been many publications concerned with the processes, methods and techniques of reverse engineeering (RE), few have taken into account experimental metrics which could provide some guides for researchers [Sne91b]. In particular, there is very little information on the effort required to execute a RE process.

A few authors have introduced some econometric information on the application of the process to real environments [Sne91a] but no-one, so far as we know, has described the experimentation and analyzed the data collected so as to enable the experience acquired to be used, at any rate as a guide, by those intending to carry out a RE process.

This paper describes a real case of study of RE and shows the data considered most useful for understanding the lessons learnt during the experience. Obviously, the data collected and the experience analyzed refer to the specific environment in which the experimentation was conducted. The authors will endeavour to generalize the results as far as possible to provide a valid guide for researchers and users of RE processes. The article describes how the experimentation was carried out, how the data were collected and validated, what analyses were made and what lessons were consequently learnt.

The case study we present refers to a project realized at Basica S.p.A, a software factory operating in Italy, and which had been commissioned by a bank.

2. Measurement and instruments

The measurements of the metrics concerning the code subjected to reverse engineering were executed automatically, with both commercially available instruments and others developed ad hoc.

The following are some examples of the measurements made on the programs:

- source lines of code: the number of lines of code after expansion of the copies, excluding the lines of comment;
- source lines of code for Procedure Divisions: lines of code contained in the Procedure Divisions;
- McCabe complexity;
- Halstead volume;
- N° of sections.

[1] This work has been partially supported by National funds MURST 40% under project "V&V in software engineering".

By program, we mean a code unit which is executed autonomously during operation of the information system.

The process data, on the other hand, were captured manually by the workers by means of the workday report form (WRF) shown in fig. 1. [Vot94]

The process data were classified on the basis of the following criteria:

- activity performed; these activities were obtained from the process model;
- program analyzed (the part of source code analyzed); according to the scheduling set out in the plan;
- output document where the information produced by the activity is collected; defined on the basis of the process baselines.

We decided to record two states a person could find himself in during execution of the activity:

a) the state of execution of the procedures according to the activities;

b) the state of suspension of these procedures

the data referred to state b) are classified as shown in fig. 1.

The person filling in the form can specify better what caused the suspension, by means of comments and references to the classifications provided on the form. As only two states can be recorded, it is possible to calculate the working time for each activity in the process but no other details. For example, it is not known whether there was reworking of a product as a result of discarding during verification. The data capture is accurate to the half hour.

This method of process data capture has been tested over some years on many projects and the staff of Basica is therefore used to this kind of data collection. The forms are filled in daily by each subject. As each worker is generally only assigned to one project, little time is required for filling in the form and the data provided is therefore reasonably complete and accurate. In any case, the forms are submitted to project monitoring and checks the consistency, completeness and comprehensibility of each form. If the form does not pass the validation phase, its compiler is interviewed to correct the data concerning him.

The data collected are recorded in a data base and then subjected to further consistency checks with other data on the same subjects, obtained from different sources. For example, the time filled in on the form is checked against the compiler's presence, recorded automatically by the presences reader.

Unfortunately, the data were collected with the sole aim of monitoring the distribution of effort over the various activities of the process. Hence, a lot of information which could have been useful for the commented analyses in this paper was lost and the latter are therefore limited to the data available.

3. The case study

The experiment described concerned the application of a RE process to some subsystems of a software system running at present in various banks in Italy. The characteristics of these subsystems are:

Operating system:	Siemens BS2000
Language	COBOL
N of programs	51
N of files	26
N of data	4,351
LOCs	404,050

The number of LOCs was calculated excluding the comments but including the copy books. The experimentation involved a group of 7 people for a total of approximately 10,000 hours of work analyzed (10 months).

The automatic tools used did not cover all the activities required by the plan and were found to be dishomogeneous. In particular, the tool used for restoration (VIASOFT tools) is not available on the BS2000 platform and it was therefore necessary to create a transference procedure and adapt the source programs analyzed to the MVS environment. It was also necessary to subject the source code to static analyses using tools built ad hoc in the UNIX (HP-UX) environment, as many utilities are available in this environment for easy processing of program texts. The information obtained with the various tools was stored in the repository from which the documentation is then produced. It had not been possible to use this tool from the very beginning of the experimentation and so a large part of the documentation was produced traditionally by means of common word processing tools.

4. The process model

The process model on which the experimentation was based includes the RE activities described in the literature [Chi90] and the restoration activities described in the continuation of this paper and in greater detail in [Vis94a].

The process model is shown in fig. 2 and briefly

outlined below.

Inventory Software System. In this phase, all the elements composing the system are individuated and the source code is prepared for analysis in the MVS environment.

The following cross references are extracted from the old software system: call dependence X-ref, copybook X-ref and file access X-ref.

Reconstruct Logical Level of Data. The hierarchical structure of the Logical Data Model is reconstructed from the data description. The aliases are recognizable entities in the applicative domain. The data are classified in three sets:

- applicative domain data: the attributes of recognizable entities in the applicative domain.
- control data: these have no correspondence with the applicative domain but are used to record the occurrence of an event during the execution of a program, so that other programs can adapt accordingly. Flags validated by one program and used by others, asynchronously, to determine their behaviour according to the previous history of the software system, are typical examples of control data.
- structural data: these are necessary for managing the organization of data bases or files. Links are typical examples.

The data in the first class compose the conceptual level [Vis93b].

Reconstruct Functional Requirements. This activity aims to acquire the Software Requirements Specifications of the system under study. The expert of the applicative domain is closely involved in this phase, which defines the expected functions in the system, i.e., those that are significant from the user's point of view.

Most of the information gained in this phase serves to produce the user documentation [Vis93b]

Abstract Data. All the data in the applicative domain which belong to the Logical Model and are not dead are associated with the corresponding meaningful concept for the applicative domain.

Reconstruct Logical Level of Programs. Each program is associated with a structure chart in which each module corresponds to a SECTION or an external subroutine of the program. Dead data (data not used by the program) and dead instructions (instructions which cannot be run) are both identified in this phase. The former are communicated to the "Abstract Data" activity while the latter are erased from the Structure Chart, which thus constitutes the logical model of the functions.

Restore Logical Model. Restoration involves introducing changes to improve the structure of the programs and make them easier to maintain, without causing repercussions on the data or interfaces with other systems. Some examples of modifications are:

- renaming variables, making their identifiers more meaningful;
- extracting modules with high internal cohesion from those with low cohesion and isolating them in the structure;
- externalizing modules which, in the current process, are in line with the main;
- localizing variables declared to be global but used locally in both existing modules and in processes extracted during restoration.

Execution of these activities is facilitated by the expected functions derived from the analysis of existing information. In fact, thanks to this knowledge, the operators can extract the functions from the modules present in the logical model. This makes the logical model more readable and its modules less complex.

Documentation Editing and Production. This phase includes the activities of final packaging of the documentation already produced in the single phases.

Test Restored System. The testing activity is carried out using the test cases which had already been used to test the original system. These cases are derived by direct capture of the transactions which were really made at an operative branch of a bank. The activity aims to demonstrate the functional equivalence of the restored program to the original one.

The process model described does not show the other activities executed, listed below:

- checking of the documents produced by each phase
- validation with experts of the applicative domain
- training of the reverse engineers in the concepts typical of the applicative domain
- other activities required for solving minor operative problems connected with the automatic tools used (e.g. the VIASOFT tools).

5. Analysis of the results

From analysis of the results we aimed to gain information on the distribution of the effort among the activities in the process model, so as to be able to forecast the effort required to carry out RE of a software system.

We can see from table 1a that about 28% of the effort is devoted to understanding the data and about 55% to restoration of the programs, the remaining

17% being spent on the other phases.

Table 1a - EFFORT DISTRIBUTION per PHASE

Inventory sw system	4 %
Test	6.5 %
Documentation	6.5 %
Reconstruct log. lev. of data	8 %
Reconstruct log.lev.of programs	18 %
Abstract data	20 %
Restore logical model	37 %

In our case study, the Reconstruction of the functional requirements phase was not carried out because at the time, the user was not interested in financing this part of the project.

The effort spent on the process can also be analyzed on the basis of the classes of activity carried out within each phase. The overall distribution among the set of phases is shown in table 1b. In the table the activities indicated as "typical phase activities" are the specific task considered in paragraph 4. The other activities indicated in the table are not specific to reverse engineering processes but can be executed in any software process.

Table 1b Distribution of effort per activity

Documentation (all phases)	14.8%
Validation	3.5%
Utility production	1.2%
Project management	1.0%
Waiting status	1.0%
System support	0.6%
Typical phase activities	77.9%

The documentation activity is partly spread over the various phases and partly concentrated in the final phase of the whole project; its high incidence is due to the fact that it is necessary to integrate the various parts of the documentation, which are realized by different tools on different platforms (pc, unix workstation, mainframe). Table 1b indicates the overall effort spent on the documentation.

The Utility production activity involves realizing and adapting some specific tools to support the analyses made in each phase, where these are not available in the case tools to be used (e.g. particular queries on the repositories).

The activity indicated as Waiting status is the one indicated on the WRF.

The Validation activity was concentrated in our case in the Abstract Data phase, in which the proposed conceptual model is validated by experts in the applicative domain.

These data show that about a quarter of the effort is spent on other activities than the ones typical of a RE process. This is partly due to the particular nature of reverse engineering activities and in this sense does not vary according to the system to be reversed but it is also due to the specific software system, the language it is written in and the environment it runs on. In fact, some platforms are better endowed with available instruments than others.

The process model we adopted led us to organize the data collected in two classes: the data relative to reverse engineering of the programs and those to reverse engineering of the data; the latter are distinct in the process as they require different skills and tools from those centred on the programs.

Table 2 illustrates the data on productivity for the activities of data redocumentation of each of the subsystems analyzed. Overall, productivity was 1.55 datum/hour.

Table 2 Effort and productivity for data reverse engineering

	No. of data	Reconstruct logical level of data		Abstract data		Total productivity
		Hours	Productivity	Hours	Productivity	
Subsytem1	1,023	185	5.5	438.5	2.3	1.6
Subsystem2	644	115	5.6	518.5	1.2	1
Subsystem3	2,684	496.5	5.4	1,045	2.6	1.7
all subsystems	4,351	796.5	5.46	2,006	2.17	1.55

Owing to the complex relationships existing among all the programs and all the files in each subsystem, it was virtually impossible to isolate the metrics for each program, so we combined them under the broad heading of the subsystems. These were sufficiently separate from the point of view of data access, i.e. most programs in each single subsystem use the same data structure.

Table 2 shows that approximately 70% of the effort spent on data redocumentation is required to realize the conceptual data model (Abstract data). This activity is particularly important in systems lacking documentation on the structure, meaning and use of the data or in which such documentation is not up to date with the real state of the system. The high level of effort is explained by the fact that activities with a high conceptual content cannot be supported by automatic tools.

From the statistical point of view, the values in table 2 cannot be subjected to analysis as the sample is too small.

Productivity for reconstructing the logical data model alone was more than double that for obtaining the conceptual model (Abstract data). This difference is due to the fact that the logical model can be obtained largely using automatic tools whereas abstraction requires the intervention of human resources. Nevertheless, data abstraction is necessary to help the final user understand the software system he is using better.

Moving on from the data to the programs, we analyzed the correlations existing between the metrics characterizing the programs and the hours spent to realize the process; the correlation coefficients are shown in Table 3.

Table 3 Correlation coefficients

	Hlog	Hr	Ht
Lo	0.53	0.64	0.69
MC	0.63	0.35	0.45
Ha	0.68	0.63	0.71
Ns	0.53	0.89	0.89
Nm	0.41	0.85	0.85

Lo = LOCs of source
Mc = McCabe complexity
Ha = Halstad volume
Ns = No. of original sections in the programs
Nm = No. of modules after restoring
Hlog = Persons Hours for reconstructing logical level of programs;
Hr = Persons hours for obtaining restored programs
Ht = Hlog + Hr

To assess the reliability of these results, we calculated the 95% confidence intervals for all the correlation coefficients; these are shown in table 4.

The values of the complexity parameters had no particular correlations with the other variables, whereas the number of modules obtained with the restoration (Nm) of the programs correlated well with the total hours of work on the programs.

The activities involved in reconstructing the logical level of programs are mainly automatic, so the complexity of the programs is resolved by the automatic tool used. This is an explanation of the low correlation existing between the complexity of the programs and the effort.

Table 4 Confidence intervals at 95% for correlation coefficients

	Hlog	Hr	Ht
LOCs source	0.3	0.44	0.51
Lo	0.7	0.78	0.81
McCabe	0.43	0.08	0.19
Mc	0.77	0.57	0.64
Halstead	0.49	0.43	0.53
Ha	0.8	0.77	0.82
n° sections	0.22	0.79	0.79
Ns	0.73	0.94	0.94
No modules aft. rest.	0.14	0.75	0.74
Nm	0.61	0.91	0.91

The high correlation between the effort for restoration (Hr) and both the number of preexisting modules (Ns) and the number obtained after restoration (Nm), derives from the need for human intervention in these activities. In particular, since the original programs are characterized by modules with low internal cohesion, the worker in this phase is obliged to consult continually with many sections to trace an expected business function, so that the effort is to some extent connected with the number of sections in the original program (Ns). Hr, on the other hand, is correlated with the number of modules obtained with the restoration (Nm), because the latter represent the functions the operator has extracted, for each of which he has had to extract the instructions that implemented these modules in the original programs, using various techniques.

The complexity of the programs was not correlated with the restoration effort because the restoration procedure decided on required very

complex program fragments to be reused without further restoration.

To assess productivity on the programs we measured the time spent on reconstruction of the logical level and restoration and compared it with the LOCs of the programs to be restored. These data are shown in table 5.

Table 5 Productivity for Reconstruct logical level and restore programs

Reconstruct logical level of programs	495.5 LOCs/hour
Restore logical level of programs	201 LOCs/hour

From the table 5, we can observe that productivity with respect to the reconstruction of the logical level alone (495.5) was more than twice that for restoration; this is again due to the fact that the activity is mainly carried out with automatic tools.

The graph in Fig.3 shows the high standard deviation from the mean. This can be attributed to the fact that the effort devoted to restoration does not only depend on the programs to be restored and the processes but also on the project manager's decisions.

In fact, in view of the data presented here, it does not seem to be possible to base the estimates of cost of realization effort exclusively on the number of equivalent LOCs as in [Jar94] but becomes necessary to consider distinct models for the data and for the programs. Nor is it possible to use the model proposed in [Sne91a], where the productivity levels are weighed against factors derived from the characteristics of the system to be reversed.

For our process, it is impossible to formulate an econometric model to forecast the restoration effort because this depends on the restoration depth the project manager wishes to obtain. Table 6 reports a few examples of productivity values for the programs before and after restoration.

As can be observed, program fc0000 features high productivity owing to the fact that the restoration was not carried out in great depth, since the mean complexity of the modules extracted remained high (53.1) and the number of modules extracted was virtually equal to the number of sections that had originally composed the program. In this case, after having individuated the expected functions in the 30 modules that had been extracted, the project manager decided that reducing the complexity of each of them would require excessive effort without sensibly improving program comprehension.

For the other programs, however, such as va0000, the work of restoration was intensive as both regards extraction (from 149 sections to 28 modules) and lowering the complexity (from 596 to a mean of 23).

6. CONCLUSIONS

This case study shows yet again that econometric models are dependent not only on the process but also on the environment it is applied in and that they cannot therefore be generalized but must be specifically derived from these.

A first general consideration is that the tools to be used in a process always require checking on the field in real cases. In fact, all the tools used, despite the numerous references to their credit and the fact that, functionally, they should have covered all the requirements of the process, were found in practice to be inadequate and needed other support tools.

Some of the negative aspects found in the tools

Tabe 6 Metrics computed on restored programs

BEFORE				AFTER				Restore productivity
Name of the original program	LOCs source	McCabe	sections	No of extracted modules	Sum of LOCs of extracted modules	Average McCabe complexity	Std. dev. of McCabe complexity	
aa0000	13,776	598	146	43	25,902	17.7	29.13	76.7
ia0000	7,951	145	34	16	6,065	10.38	15.98	77.6
nm0000	5,421	24	5	3	498	6.67	5.51	151.6
va0000	13,794	596	149	28	17,780	23	34.98	74.6
fc0000	6,896	95	27	30	4,603	53.1	7.38	1532.4

WORK DAY REPORT FORM

PROJECT _____

NAME _____ DATE _____

Phase code	Hours	Status	INPUT object analyzed	OUTPUT object produced	comments

PHASES

1. Inventory software system
2. Reconstruct logical level of data
3. Abstract data
4. Reconstruct functional requirements
5. Reconstruct logical level of programs
6. Restore logical model
7 Test and debug
8 Produce documentation

STATUS

E. Execution
W Wait
- W1 Wait for automated task execution
- W2 Wait for the completion of other activities
- W3 Communication problems
- W4 Hardware problems
- W5 Software problems
- W6 Resource problems

Fig. 1 The sample work day report form

refer to their capacity to handle software systems of real dimension, e.g. those with over 1000 data and 8000 lines of code. Waiting time for the response to queries or the extraction of modules became particularly long.

It is therefore wise to ensure that the automatic tools chosen for adoption can be adapted to the requirements of the environment they will be used in.

Another general consideration regards the automization of the reverse engineering process. In the literature, some well formalized processes have appeared, which enable the conceptual model of a program to be derived automatically, while tools are available on the market which offer highly refined functions for automizing the abstraction processes. Our experience has taught us, however, that these automatically derived representations with a high level of abstraction retain all the comprehension problems which characterized the source code.

Hence, when starting from a poorly readable program, human intervention is necessary after the automatic process to carry out the operation described in our process as "restore logical level", which serves to improve the structure of the programs and make them more comprehensible.

An innovative finding of our analysis is that although the human intervention required to improve the comprehensibility of the programs can notably reduce productivity, it is possible to organize the process so that the effort spent on this improvement is anchored to the budget and time available. This implies that if a sufficient budget is not available, programs of poor quality which are not useful enough to justify restoration should not be subjected to this further phase of the reverse engineering process.

To complete the picture, it must be pointed out that the effort spent on the reverse engineering process is repaid by the consequent economical management of the application, as illustrated in detail in [Vis94b].

Many types of data on real case studies may be collected and many other results obtained from analysis but, as is well known, the data collection should be oriented towards more specific objectives.

As stated in paragraph 2, highly percise data are needed in order to obtain the maximum information on what events occurred during the execution of a project. For this purpose, we need to be able to rely on tools

supporting the formalization of the RE process and its enactement. These would enable more reliable information and greater detail to be gained. The data collected in this fashion would help to make more specific analyses aiming to solve the problems of assessment of the process characteristics

For the above reason, future work in this field will aim to introduce process formalization and enactement tools for automatic data collection and analysis of the data collected using the GQM model [Bas92]. Highest priority will be given to analyses providing econometric assessment of the RE process.

REFERENCES

[Ant94] Antonini P., Canfora G., Cimitile A. "Re-engineering legacy systems to meet quality requirements: an experience report". 16th Int. Conf. on Sw. Eng. Sorrento May, 16 1994 - IEEE Comp. Soc. Press

[Arn94] Arnold R. "Software Reengineering - Tutorial" 16th Int. Conference on Software Engineering, Sorrento May, 16 1994 - IEEE Computer Society Press

[Bas84] Basili V. R., Weiss D. M. "A methodology for collecting valid software engineering data" IEEE Transactions on Software Engineering Vol. 10 No. 6 Nov. 1984.

[Bas92] Basili V. R., "Software modeling and measurement: the Goal/Question/Metric paradigm" Computer Science Technical Report Series CS-TR-2956 University of Maryland UMIACS-TR-92-96 - September 1992

[Chi90] Chikofsky E. J., Cross II J.H. "Reverse engineering and design recovery: a taxonomy" IEEE Software Jan. 1990.

[Deb92] Debest X. A., Knoop R., Wagner J. "Reveng: a cost-effective approach to reverse-engineering"

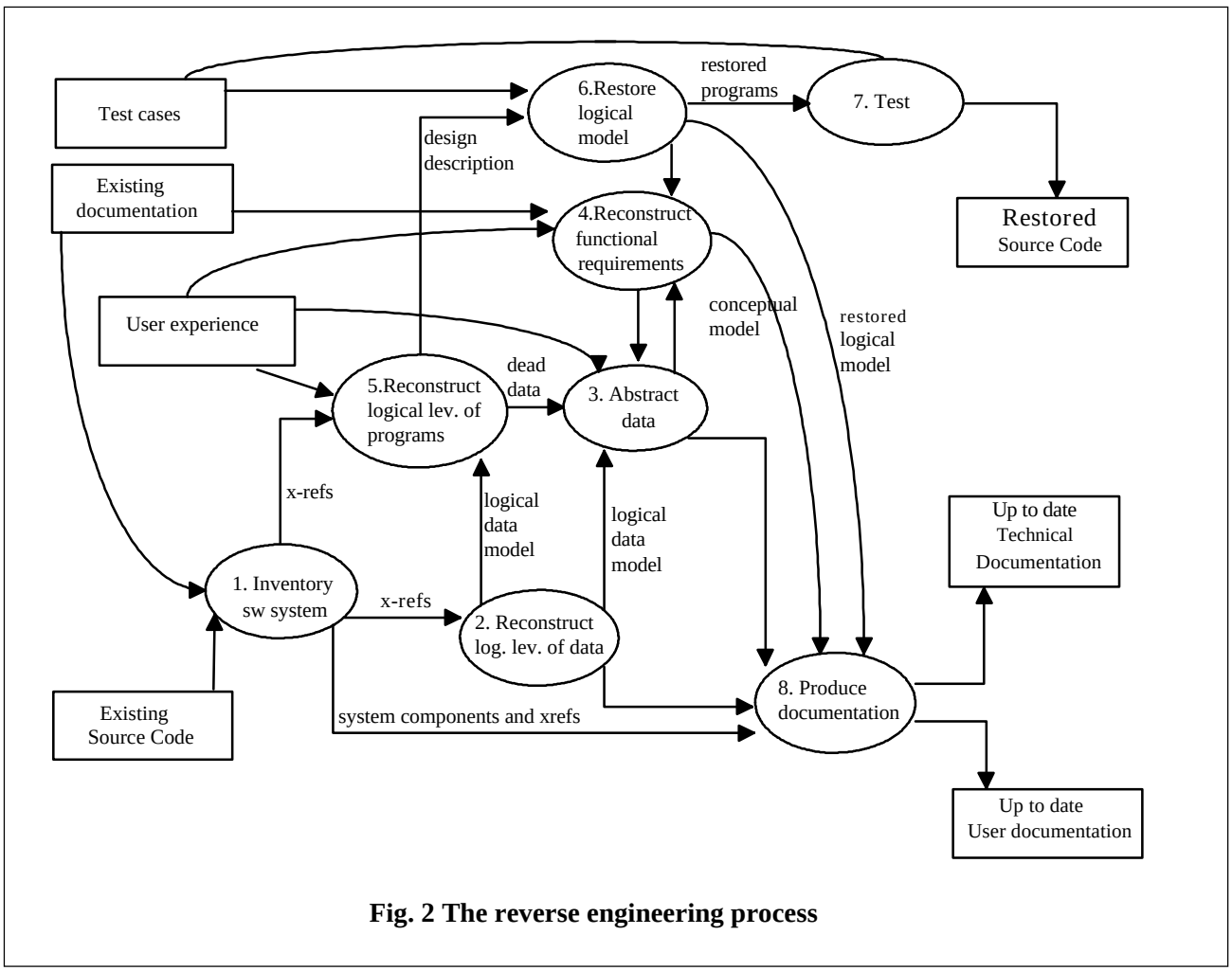

Fig. 2 The reverse engineering process

Software Engineering Notes Vol.17 No. 4 Oct. 1992

[Jar94] Jarzabek S., "Life-Cycle approach to strategic Re-engineering of software", Journal of Software Maintenance: research and practice Vol. 6 No. 6 1994

[Lap83] Lapin L. L., "Probability and statistics for modern engineering", PWS Publishers - B/C Engineering division - Boston 1983.

[Mar94] Markosian L., Newcomb P., Brand R., Burson S. and Kitzmiller T. "Using an enabling technology to reengineer legacy systems". Comm. of the ACM Vol. 37 no. 5. May 1994

[Sne91a] Sneed H. M. "Economics of Software Re-engineering" , Journal of Software Maintenance: Research and Practice Vol. 3 no. 3. Sept. 1991

[Sne91b] Sneed H. M. "Bank Application reengineering & conversion at the Union Bank of Switzerland". Conference on Software Maintenance 1991. Sorrento Italy. IEEE Comp. Soc. Press.

[Vis93a] Visaggio G., Cutillo F. and Fiore P. "Identification and extraction of domain indipendent components in large programs" Proceedings of the Working Conference on Reverse Engineering, Baltimora 1993.

[Vis93b] Visaggio G, Abbattista F., Lanubile F., "Recovering Conceptual Data Models is Human Intensive". The 5th International Conference on Software Engineering and Knowledge Engineering. 1993 Knowledge System Institute

[Vis94a] Visaggio G., Abbattista F., Fatone G.M.G., Lanubile F. "Analyzing the application of a reverse engineering Process to a real situation". 3rd Workshop on Program Comprehension Nov 14-15 1994 Washington D.C. USA. IEEE Computer Society Press.

[Vis94b] Visaggio G., "Process improvement through data reuse", IEEE Software Vol.11 No.4, July 1994

[Vot94] Votta L. G., Perry D. E., Staudenmayer N. A. "People, organization, and process improvement" IEEE Software July 1994.

[Wol93] Wolf A. L., Rosenblum D. S. "A study in software process data capture and analysis" Proceedings of the 2nd International Conference on the Software Process (ICSP2), Berlin 1993. IEEE Comp. Soc. Press.

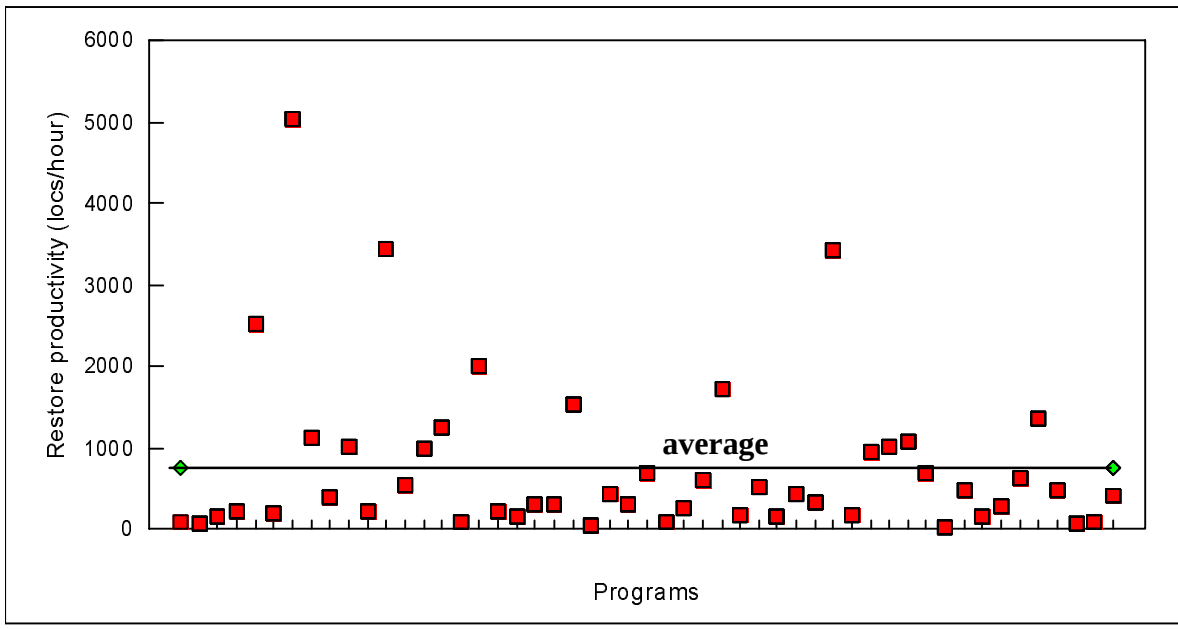

Fig.3 Productivity for restoring logical level