An Empirical Study of Reuse, Quality, and Productivity

Bill Frakes

Department of Computer Science
Virginia Tech
wfrakes@vt.edu

Giancarlo Succi

Department of Electrical and Computer Engineering
University of Calgary
Giancarlo.Succi@enel.ucalgary.ca

Abstract

This paper presents an analysis of four sets of industrial data to determine if software reuse is correlated with higher levels of software quality and productivity.

1. Introduction

Systematic software reuse is a goal for many organizations today because they believe that higher levels of reuse will significantly improve software quality and productivity[Frakes, 1994]. While this belief is widespread, there has been insufficient empirical validation of it, though [Melo et al., 1996] have recently published some related work. Good general introductions to software metrics may be found in [Conte et al., 1986] and[Fenton et al. 1997].

This paper presents an analysis of several sets of industrial data to determine if reuse is in fact correlated with higher levels of software quality and productivity.

The specific research questions we will examine, framed as null hypotheses, are:

1. There is no relationship between software quality, measured by counting errors, and amount of reuse as measured by reuse level RL and reuse frequency RF.

2. There is no relationship between perceived software quality, measured by asking the developers of the code to rate module quality on a 10 point scale, and amount of reuse as measured by reuse level RL and reuse frequency RF.

3. There is no relationship between software productivity, measured as the effort it takes to develop modules, and amount of reuse as measured by reuse level RL and reuse frequency RF.

4. There is no relationship between software productivity, measured as the number of lines of code produced per unit effort, and amount of reuse as measured by reuse level RL and reuse frequency RF.

This paper is organized as follows. Section two describes amount of reuse metrics, how RL and RF are calculated, and the automated tool we used to calculate them for C and C++ software. Section three contains the results of our correlation analysis of RL, RF, and quality and productivity variables.

Section four analyses the results. In section five we present a summary of our conclusions and discuss future work.

2. Measuring Amount of Reuse

The amount of reuse is the quantity of reuse for a lifecycle product. For example, if we have a system composed of 100 functions, we might ask how many of those functions are new, and how many reused. If 10 of the functions are reused we might say that the amount of reuse in the system is 10%. However the situation is seldom straightforward. The amount of reuse can be divided into internal and external[Frakes, 1990]. Internal is the reuse of components from the same system, external is the reuse of external components, that is those developed outside the system project. Both of them should be considered when defining the amount of reuse metrics.

Several amount of reuse metrics have been proposed. An extensive survey of them is in [Frakes, 1996]. These metrics are complementary since they describe different aspects of the amount of reuse. In this paper we will use RL (reuse level) and RF (reuse frequency).

2.1. RL and RF

Reuse level measurement assumes that a system is composed of parts at different levels of abstraction. The levels of abstraction must be defined to measure amount of reuse. For example, a system written in C is composed of modules (.c files) that contain functions, and functions that contain lines of code. The amount of reuse in a C based system, then, can be expressed in terms of modules, functions, or lines of code. A software component may be internal or external. An internal lower level component is one developed for the higher level component. An external lower level component is used by the higher level item , but was created for a different higher level item or for general use.

The following quantities can be calculated given a higher level item composed of lower level items:

- L: the total number of lower level items in the higher level item.
- E: the number of lower level items from an external source in the higher level item.
- I: the number of lower level items in the higher level item which are not from an external source.
- M: the number of items not from an external source which are used more than once.

These counts are of unique items (types) not tokens (references).

Given these quantities, the following reuse level metrics can be defined:

External Reuse Level: E / L
Internal Reuse Level: M / L
Total Reuse Level: External Reuse Level + Internal Reuse Level

Internal, external, and total reuse level values are between 0 and 1. A reuse level of 1 indicates complete reuse. A reuse level of 0 indicates no reuse.

To calculate these reuse measures a user must define the abstraction hierarchy, a definition of external repositories, and a definition of the "uses" relationship. For each component, we must know the name of the component, source of the component (internal or external), the level of abstraction, and the amount of usage.

The variables and reuse level metrics are:

ITL: internal threshold level, the maximum number of times an internal item can be used before it's considered reused.
ETL: external threshold level, the maximum number of times an external item can be used before it's considered reused.
IU: number of internal lower level items which are used more than ITL.
EU: number of external lower level items which are used more than ETL.
T: total number of lower level items in the higher level item, both internal and external.
Internal reuse level: IU / T
External reuse level: EU / T
Total reuse level: (IU + EU) / T

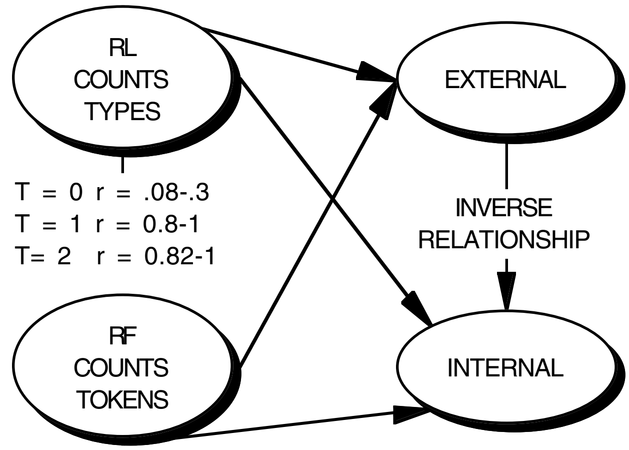
The *reuse frequency* metric is based on references (tokens) to reused components rather than counting component types as was done for reuse level. The variables and reuse frequency metrics are:

- IUF: number of references in the higher level item to reused internal lower level items.
- EUF: number of references in the higher level item to reused external lower level items.
- TF: total number of references to lower level items in the higher level item, both internal and external.
- Internal reuse frequency: IUF / TF
- External reuse frequency: EUF / TF
- Total reuse frequency: (IUF + EUF) / TF

Program size is often used as a measure of complexity The complexity weighting for internal reuse is the sum of the sizes of all reused internal lower level items divided by the sum of the sizes of all internal lower level items within the higher level item. An example size weighting for internal reuse in a C system is the ratio of the size (calculated in number of lines of non-commentary source code) of reused internal functions to the size of all internal functions in the system.

The software tool *rl* calculates reuse level and frequency for C code. Given a set of C files, rl reports the following information:

1. internal reuse level
2. external reuse level
3. total reuse level
4. internal reuse frequency
5. external reuse frequency
6. total reuse frequency
7. complexity (size) weighting for internal functions



(T is threshold level, r is correlation value)

Figure 1 - Relationships of RL and RF

The user may specify an internal and external threshold level. The software allows multiple definitions of higher level and lower level abstractions. The allowed higher level abstractions are system, file or function. The lower level abstractions are function and NCSL (Non-Commentary Source Lines of code) The *rl* tool is available via anonymous ftp at ftp.cs.vt.edu, directory pub/reuse.

2.3. Relation of RL and RF

The general relationships of RL and RF, both internal and external, at different threshold levels is shown in Figure 1. In general RL is based on counting types and RF is based on counting tokens. External versions of RL and RF are inversely related to the internal versions. RL and RF at threshold 0 have low correlations (r in the diagram--T is threshold level; these data come from the Software Uno dataset, as described later in the paper). This indicates that they are measuring different things. At higher thresholds, however, they are highly correlated (in the 0.8 to 1.0 range) and therefore appear to be measuring the same thing.

We will use the following notation for the RL and RF metrics in the rest of this paper.

IRLn is the Internal Reuse Level at threshold n

ERLn is the External Reuse Level at threshold n

3. Correlations of RF and RL with Quality and Productivity

We analyzed software from four industrial sites to determine the relationships between the amount of reuse, as measured by reuse level and reuse frequency, and quality and productivity. Software quality is often measured using counts of software errors. Module deltas are sometimes used when error densities are not available, since they correlate highly with them. Another way to evaluate quality is to use subjective judgements of the module developers. In this study, developers rated the quality of the modules on a scale of 1 (worst) to 10 (best). Productivity is often measured as the amount of code produced per unit of time, and we have followed this method in the present study.

Table 1 summarizes the data for the four data sets used in the study. An "X" in a cell in Table 1 indicates that a given quality or productivity measure was used for a given data set.

Data Set	Quality			Productivity	
	Error density	Evaluation	Module Deltas	Time	SLOC/Time
PRC			X		
Consultant A		X			
Consultant B	X	X		X	X
Software Uno				X	X

Table 1: Quality and Productivity Measures in this Study

3.1. PRC Inc. Data

Our first data set was obtained from PRC Inc., a large government contracting company located in Fairfax, Virginia USA. The domain of the software was text processing and retrieval. . We do not have demographic data on the developers of this software. ERL and ERF were measured at threshold level zero for three software modules written in the C language. The number of module deltas were also measured and used as quality indicators. Deltas were calculated using SCCS, a change management tool on UNIX. Table 2 summarizes these measures

	ERL 0	ERF 0	Module Deltas
Module 1	0.47	0.4	13
Module 2	0.44	0.44	8
Module 3	0.33	0.25	16

Table 2: PRC reuse and module delta data

The plots in Figure 2 show the relationships between module deltas and ERF0 and ERL0. These data indicate a negative correlation between external RF and RL, and module deltas. This would be expected because higher levels of reuse would be expected to correlate with fewer faults, since reusable components are expected to have been tested more thoroughly, or at least to have had longer use allowing errors to be identified an corrected. The Spearman correlations for ERL0 and module deltas is 0.5 and for ERF0 and module deltas it is 1.0. The small number of data points did not warrant an inferential analysis of the correlations.

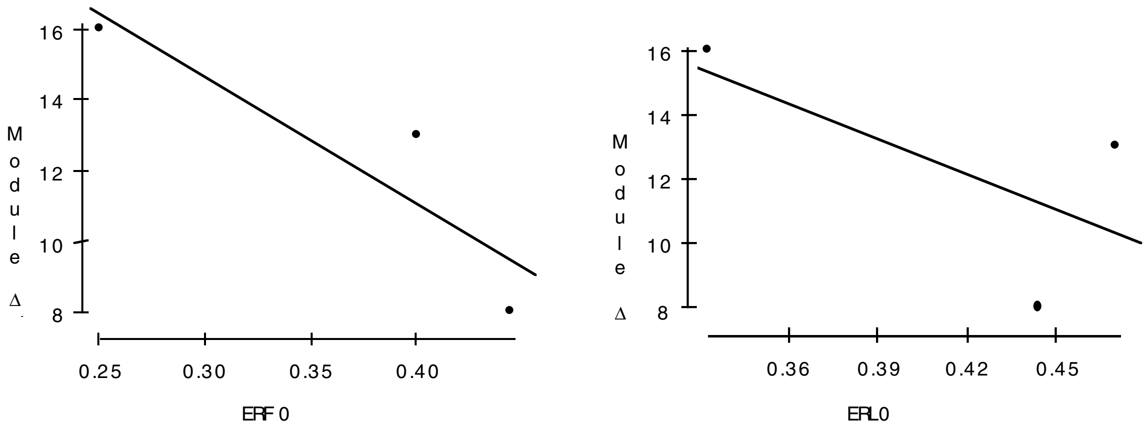


Figure 1: Correlation between ERF0 and Module Deltas at PRC

3.2. Consultant A Data

Consultant A data are based on software developed for a medium sized software organization located in northern Italy. The company allowed us to use the code on condition of anonymity. The domain of the software is a graphical user interface written in the C++ language under HP/UX version 9. The developer has a B.S. in computer science and about two years experience.

Table 3 describes the 6 data points we collected from Consultant A, and summary statistics of that data. Modules varied in size from 442 to 2543 NCSL, with a median value of 2045.5. RL and RF data were calculated on the C code generated by the C++ compiler, since the *rl* tool does not handle C++ directly. Quality was measured by having the developer rate the quality of modules on a 10 point scale (10 = best). Quality ratings varied from 6-8, with a median rating of 7.

	ERF0	**ERL0**	**IRF1**	**IRL1**	**NCSL**	**Quality**
Module 1	0.39	0.44	0.20	0.04	2055	8
Module 2	0.34	0.41	0.36	0.10	1860	7
Module 3	0.21	0.46	0.76	0.35	2543	7
Module 4	0.19	0.20	0.57	0.30	442	6
Module 5	0.27	0.44	0.58	0.14	2348	7
Module 6	0.24	0.43	0.65	0.21	2036	8
Mean	0.27	0.40	0.52	0.19	1880.67	7.17
Median	0.26	0.43	0.57	0.17	2045.50	7
Std. Dev.	0.08	0.10	0.20	0.12	745.89	0.75
Min.	0.19	0.20	0.20	0.04	442	6

Max.	0.39	0.46	0.76	0.35	2543	8
25th%ile	0.21	0.36	0.32	0.09	1505.50	6.75
75th%ile	0.35	0.44	0.68	0.31	2396.75	8

Table 3 - Consultant A Data

We analyzed the relationship of perceived quality and reuse level using Spearman rank correlations. There is positive correlation (0.27) between ERL0 and quality, and a stronger positive correlation (0.53) between ERF0 and perceived quality. Neither of these correlations, however, is statistically significant. This is most likely because of the small sample size.

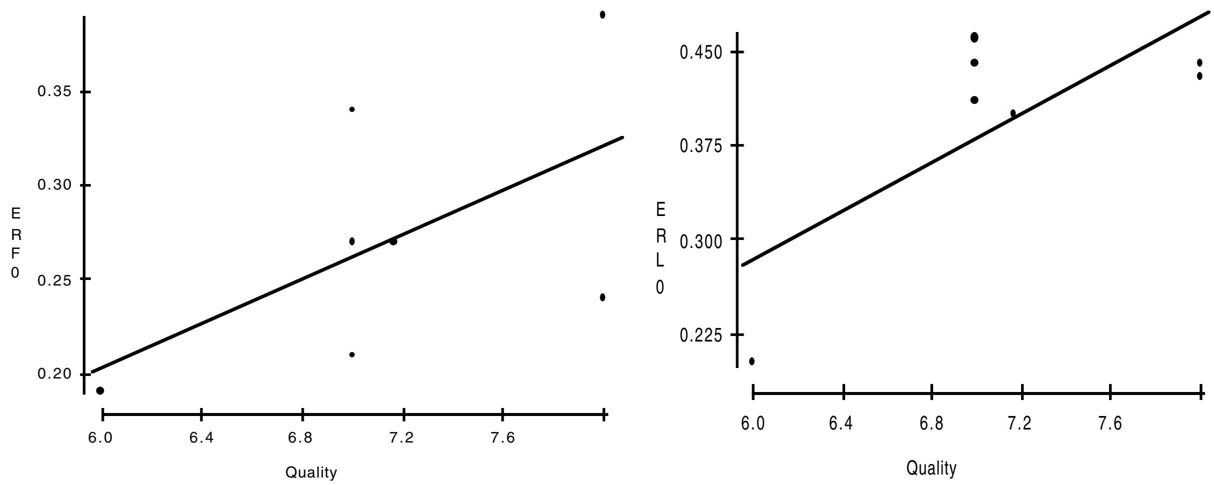


Figure 3 - Correlation of Quality to ERF0 and ERL0, Consultant A Data

3.3. Consultant B data

These data were collected from a software engineer at a telecommunications company in Europe. The software engineer has an MS degree in Electrical Engineering with 5 years of industrial experience. These data were collected on 12 C++ modules for a distributed reuse library developed under HP/UX version 9. The data collected for these modules and summary statistics are in Table 4. Modules varied in size from 50 to 3197 NCSL, with a median value of 679. RL and RF data were calculated on the C code generated by the C++ compiler, since the *rl* tool does not currently handle C++ directly. Quality was computed by having the developer rate the quality of modules on a 10 point scale (10 = best). Quality ratings varied from 6-10, with a median rating of 7. The developer kept a log of the errors he found and the time he took, in working days, to develop each file. Productivity was measured by dividing the size of each module by the time taken to produce it.

Name	Effort / Module	Versions	Quality	Bugs	NCSL	ERF0	ERL0	SLOC / Effort
Module 1	3	2	7	0	478	0.25	0.25	159.33
Module 2	6	3	7	0	2586	0.26	0.30	431.00

Module 3	10	5	8	2	662	0.24	0.43	66.20
Module 4	2	1	8	0	553	0.42	0.48	276.50
Module 5	10	3	6	3	395	0.11	0.23	39.50
Module 6	7	2	7	0	2723	0.18	0.32	389.00
Module 7	10	3	8	1	676	0.25	0.33	67.60
Module 8	15	3	7	2	3197	0.22	0.33	213.13
Module 9	7	2	7	1	2516	0.28	0.30	359.43
Module 10	5	2	6	2	682	0.20	0.34	136.40
Module 11	1	1	10	0	50	0.43	0.43	50.00
Module 12	3	1	9	0	2293	0.31	0.33	764.33
Mean	6.58	2.33	7.50	0.92	1400.91	0.26	0.34	246.04
Median	6.50	2.00	7.00	0.50	679.00	0.25	0.33	186.23
St. Dev.	4.12	1.15	1.17	1.08	1144.69	0.09	0.074	213.83
Min	1.00	1.00	6.00	0	50.00	0.10	0.23	39.50
Max	15.00	5.00	10.00	3	3197.00	0.43	0.48	764.33
25th%ile	3.00	1.25	7.00	0	496.75	0.20	0.29	66.55
75th%ile	10.00	3.00	8.00	2	2568.50	0.30	0.41	381.60

Table 4: Consultant B data

We found a significant positive correlation (0.76, p<.05) between perceived quality and RF0 . The plot of this data is in figure 4.

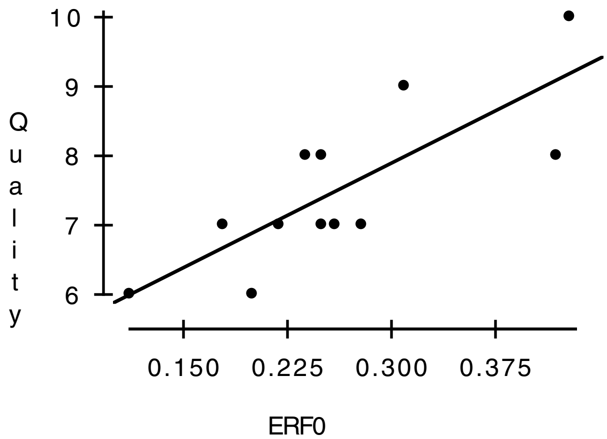


Figure 4 - Correlation of Quality and ERF0 for Consultant B data

There are significant negative correlations between RF0 and fault densities and between RF0 and development time in person days per module. Plots of this data demonstrating these correlations can be found in Figure 5.

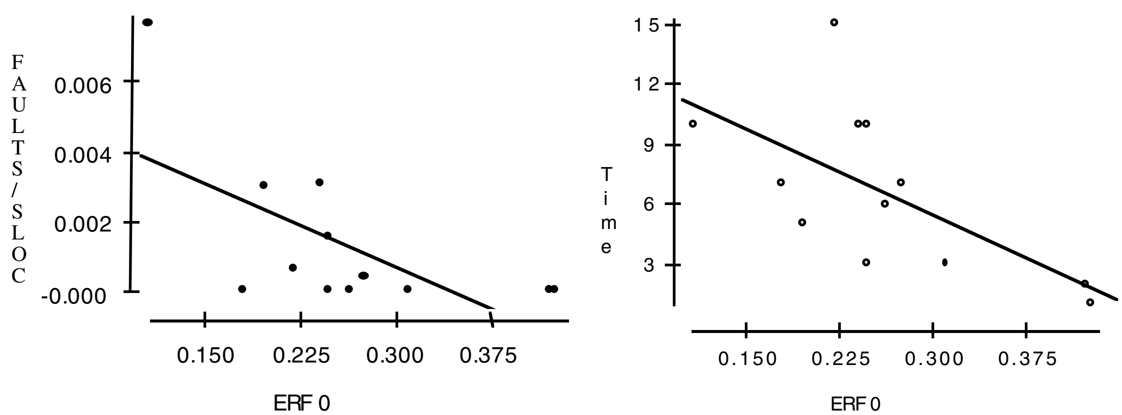


Figure 5: Plots of ERF0 vs. Faults/SLOC and Time for Consultant B Data

3.4. Software Uno

These data were collected at Software Uno S.r.l., a small software company located in Padova, Italy. The company has around 20 employees. There were four people in the team that developed the software we measured. Three of them had BS degrees and 7 years of experience, and one an MS with

2 years of experience. The programming language used was C under MS/DOS. This data was collected on 16 files from a system for the management of a hospital. The data collected for these modules, and descriptive statistics are in Table 7. Modules varied in size from 46 to 805 NCSL with a median value of 143.5. ERL and ERF data were calculated on the C code. Software Uno kept a record of the effort spent per file. Productivity was measured by dividing the size of each file by the effort, in hours, to produce it.

The Spearman rank correlation for time and ERL0 is -.448, which is not statistically significant. The correlation between productivity and ERL0 is .311, which is not statistically significant.

Name	Effort	NCSL	ERF0	ERL0	IRF1	IRL1	SLOC/ Effort
Module 1	4	98	0.010969	0.363636	0.981718	0.272727	24.5
Module 2	23	304	0.263158	0.346154	0.480263	0.153846	13.22
Module 3	4	145	0.026549	0.424242	0.943363	0.060606	36.25
Module 4	5	327	0.014337	0.315789	0.969534	0.210526	65.4
Module 5	3	120	0.016187	0.388889	0.967626	0.111111	40
Module 6	2	111	0.009259	0.416667	0.97963	0.083333	55.5
Module 7	3	46	0.003597	0.4	0.994604	0.4	15.33
Module 8	45	805	0.055965	0.37037	0.908689	0.185185	17.89
Module 9	48	749	0.045455	0.295455	0.929619	0.318182	15.6
Module 10	3	105	0.0161	0.315789	0.9678	0.210526	35
Module 11	10	113	0.005545	0.230769	0.977819	0.076923	11.3
Module 12	5	248	0.025042	0.289474	0.93823	0.131579	49.6
Module 13	4	142	0.021277	0.357143	0.952128	0.107143	35.5
Module 14	7	197	0.035061	0.3	0.934451	0.2	28.14
Module 15	3	106	0.015901	0.227273	0.962898	0.227273	35.33
Module 16	92	467	0.027735	0.28125	0.946071	0.1875	5.08
Mean	16.31	255.19	0.0370	0.3327	0.9271	0.1835	30.23
Median	4.50	143.50	0.0187	0.3310	0.9575	0.1863	31.57
Std. Dev.	24.97	230.87	0.0619	0.0607	0.1213	0.0923	17.06
Min	2	46	0.0036	0.2272	0.4803	0.0606	5.080
Max	92	805	0.2631	0.4242	0.9946	0.4000	65.40
25th%ile	3	107.25	0.0118	0.2909	0.9354	0.1081	15.40
75th%ile	19.75	321.25	0.0332	0.3842	0.9757	0.2231	39.06

Table 5: Summary of Software Uno Data

4. Evaluation of the Data

Table 6 presents a summary of the data analyses.

Data set	Domain	Data points	Variables	Results
PRC	Text Retrieval	3	ERF0 vs. Module Deltas	Negative Correlation
Consultant A	User Interface	6	ERF0, ERL0 vs. Perceived Quality	Positive Correlation
Consultant B	Distributed Repository	12	ERF0 vs. Fault Density	Significant Negative Correlation
			ERF0 vs. Perceived Quality	Significant positive correlation
			ERF0 vs. Time	Significant Negative Correlation
Software Uno	Medical Records	16	ERF0 vs. Productivity	Negative Correlation
			ERF0 vs. Time	Negative Correlation

Table 6: Summary of the results

The answers, based on our analysis of the data, to the specific research questions, framed as null hypotheses, of this study are as follows.

1. There is no relationship between software quality, measured by deltas or fault density, and reuse level as measured by external reuse frequency ERF at a threshold of 0.

Based on the data, it looks like this null hypothesis should be rejected. Both the PRC and consultant B data showed a negative correlation between quality measures and reuse level. A very small sample size precluded an inferential test of the correlation for the PRC data. In the case of the consultant B data, however, the correlation is statistically significant.

2. There is no relationship between perceived software quality, measured by asking the developers of the code to rate module quality on a 10 point scale, and reuse level as measured by reuse level ERL and reuse frequency ERF.

Based on the both consultant A and B data, this null hypothesis should be rejected. There does appear to be a positive relationship between perceived quality and reuse level.

3. There is no relationship between software productivity, measured as the effort it takes to develop modules, and reuse level as measured by reuse level ERL and reuse frequency ERF.

Our results here are inconsistent. For the consultant B data, we found a significant negative correlation between the effort to develop a module and reuse level. For Software Uno data, however, there is a positive correlation. There is a difference in the programming language: consultant B uses C++, while

Software Uno C; in this sense we might guess that C++ provides a better support to software reuse than C; this is not a strong argument, anyway.

4. There is no relationship between software productivity, measured as the amount of software produced per unit time, and reuse level as measured by reuse level ERL and reuse frequency ERF.

We found no significant linear correlation between productivity and reuse level for either of the data sets we examined.

5. Conclusions

In this paper we have explored the effect of software reuse on quality and productivity by looking at four sets of industrial software data. These data show that more reuse corresponds to higher quality ratings, and lower levels of defects and deltas. The data also shows that modules that incorporate more reusable components take less time to develop.

As to future work, though the difficulties of obtaining industrial software data are well known, without such data it will be difficult to verify the effects of reuse on quality and productivity across the software industry. More empirical studies of reuse, quality, and productivity, such as this one, need to be done to provide a solid foundation for industrial practice.

Acknowledgments

We thanks Prof. G. Kovacs for his insightful comments, N. Scioscia and G. Cardino for their help in finding data, CIM-EXP (Budapest, Hungary), PRC (Falls Church, VA), and Software Uno S.r.l. (Padova, Italy) for the data they provided us. Thanks also to Gabriella Belli, Omar Alonso, and Gloria Hasslacher for their reviews of a draft of the paper.

References

[Conte et al., 1986] Conte, S. D., H. E. Dunsmore, V. Y. Shen. Software engineering metrics and models. Benjamin Cummings, 1988
Humphrey, W. S. A discipline for software engineering. Addison-Wesley, 1995
[Fenton et al. 1997] Fenton, N. E., S. L. Pfleeger. Software metrics: a rigorous and practical approach. International Thomson Computer Press, 1997.

[Frakes, 1990] Frakes, W. (1990). An Empirical Framework for Software Reuse Research *Proceedings of the Third Annual Workshop on Software Reuse* (Technical Report No. 9014). Syracuse University CASE Center.

[Frakes, 1994] Frakes, W., & Isoda, S. (1994). Success Factors of Systematic Reuse. *IEEE Software*, 11(5), 14-19.

[Frakes, 1996] Frakes, W., & Terry, C. (1996). Software Reuse Models and Metrics. *ACM Computing Surveys, 28*(2), 415-435.

[Melo et al., 1996] Melo, W. L., L. C. Briand, V. R. Basili. Measuring the Impact of Reuse on Quality and Productivity in Object Oriented Systems. Communications of the ACM, 39(10) (Oct. 1996), 104-116