Predictive Software Measures based on Z Specifications – A Case Study

Andreas Bollin
Software Engineering and Soft Computing
University of Klagenfurt
Klagenfurt, Austria
Andreas.Bollin@aau.at

Abdollah Tabareh
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
tabareh@gmail.com

Estimating the effort and quality of a system is a critical step at the beginning of every software project. It is necessary to have reliable ways of calculating these measures, and, it is even better when the calculation can be done as early as possible in the development life-cycle.

Having this in mind, metrics for formal specifications are examined with a view to correlations to complexity and quality-based code measures. A case study, based on a Z specification and its implementation in *ADA*, analyzes the practicability of these metrics as predictors.

1 Introduction

Recent studies in the areas of software metrics and project management have stimulated a lot of ideas of how development effort can be estimated and which metrics are of relevance [9, 15, 19]. Basically, they all suggest that the collection of data and the estimation process should be performed as early and as objectively as possible – so why not taking a closer look at properties of formal specifications?

To the best of our knowledge, the only publicly available case study that took a closer look at correlations between specifications and implementations was conducted by Samson, Nevill and Dugard in 1987 [17]. The authors used Modula-2 modules and a HOPE specification to show that there is a correlation between the number of equations in HOPE and the number of lines of source code and cyclomatic complexity in the Modula-2 modules. However, the authors admit that the study is relatively small-scale as their data is based on only 9 experimental subjects.

The objective of this paper is now to shed some more light onto the question whether specifications' properties can help predicting attributes of derived implementations or not. For this, the following strategy is pursued: firstly, based on a set of well-known measures, it tries to find out whether some of the measures are correlated or not. Secondly, it suggests a prediction model for some of the measures. A case study, based on the specification and implementation of the *Tokeneer* system [5] forms the basis for these considerations. It takes the Z specification of the system and its implementation in *ADA* as the point of departure and identifies those parts of the code that unambiguously implement specific parts of the specification. After that, it calculates size, structure and quality related measures for both of the documents. Finally, it looks for correlations between the measures, and, based on the findings, it calculates a prediction model for several *ADA*-based size- and complexity-related measures.

This paper is structured as follows: Section 2 briefly introduces the code and specification measures that are used in the study. Section 3 presents the setting of the study, the experimental subject and the statistical tests used. Next, Section 4 presents and discusses the results of the correlation tests, and Section 5 presents the prediction model. Section 6 discusses possible threats to validity, and, finally, Section 7 summarizes the findings and discusses possible steps to be done next.

Submitted to:
WS-FMDS 2012

2 Measures

This section introduces the measures used for assessing the *Z* specification and its implementation in *ADA*. Please note that, due to limitations of space, only a brief overview of the measures is provided[1].

2.1 Code-based Measures

The implementation language of the *Tokeneer* specification [5] is *ADA*. In his master thesis, Tabareh [20] took a look at currently available environments that are able to generate practical measures from *ADA* code. He suggests to apply the *Understand* tool and uses the following measures (where M denotes either an *ADA* function or a procedure)[2] for a preliminary study comparing ADA and Z-based measures:

- CountLine $CL(M)$. It counts the number of physical lines.
- CountLineCode $CLC(M)$. It counts the number of lines that contain source code.
- CountLineExecutable $CLCE(M)$. It counts the number of lines containing executable *ADA* code.
- CountLineCodeDecl $CLCD(M)$. It count the number of lines containing declarative *ADA* code.
- Knots Count $KNOTS(M)$: It is a measure for the structuredness of a module and counts overlapping jumps in the program flow graph.
- Cyclomatic Complexity $CYC(M)$. It measures the maximum number of linearly independent paths through a program and is extracted by counting the minimum set of paths which can be used to construct all other paths through the graph.

In order to focus even more on structural properties of the code, this study additionally makes use of Shepperd and Ince's Information flow count [18]. The general idea is that the complexity of a module is related to the number of flows or channels of information between the module and its environment. For this, the *Understand* tool can be used to generate the call-graph, and the flow of data and control then forms the basis for the calculation of the Sheppard Information Flow *SI* of a module M:

- Fan-in ($FIN(M)$): It comprises the number of data-flows terminating at a component M.
- Fan-out ($FOUT(M)$): It comprises the number of data-flows originating at a component M.
- Information Flow($SI(M)$): It comprises the number of information flows related to a component M and is calculated via $(FIN(M) * FOUT(M))^2$.

2.2 Specification Measures

Most of the complexity measures for formal specifications focus on size. The reasons are that size-based measures (like lines of specification text) are easy to calculate and yield a single number that is easy to interpret. This is not so much the case for structure- and quality-related measures. Their calculation is usually based on the notion of control and data dependencies, concepts that are not necessarily dominant principles of a specification language. However, several authors [4, 12, 13] demonstrated that a reconstruction of these dependencies is possible.

Recently, Bollin showed that coupling and cohesion based measures can reasonably be mapped to formal Z specifications [3]. The basis for the calculation of all the measures is a graph that contains vertices (called *prime*s) for all predicates and declarations of the specification, and arcs representing (reconstructed) control and data dependencies [2]. With such a graph as a basis, the following measures (defined for schemas ψ that are part of a specifications Ψ) are used in the remainder of this work:

[1] An in-depth discussion of other specification-based measures can be found in the Ph.D. thesis of Bollin [1].

[2] The tool and a description of the measures can be found at the Understand homepage at www.scitools.com. Page last visited: May 2012.

- Conceptual Complexity $CC(\psi)$: The conceptual complexity equals the total number of prime vertices in the graph (representing a schema ψ).
- Logical Complexity $v'(\psi) = (l,u)$: The lower bound value l of the measure is 1 plus the number of primes that are terminal vertices of control dependency arcs. It can be compared to counting the number of decision statements in programs. The upper bound value u equals 1 plus the total number of control dependencies. It reflects the total amount of dependencies to be considered.
- Definition-Use Count: $DU(\psi)$: This measure equals the total number of data dependencies.
- Use Count $USE(\psi)$: The use count equals the number of identifiers used in the schema ψ.
- Definition Count $DEF(\psi)$: The definition count equals the number of identifiers referring to an after state in the schema ψ of the specification.
- And-Count $AND(\psi)$: This measure equals the number of AND-combined predicates in ψ.
- Or-Count $OR(\psi)$: This measure equals the number of OR-combined predicates in ψ.

Semantics-based measures can be calculated by generating slices. The idea goes back to the work of Weiser [21] who introduced five slice-based measures for cohesion: Tightness, Coverage, Overlap, Parallelism and Clustering. Ott and Thuss [14] partly formalized these measures. Coupling, on the other hand, was originally defined as the number of local information flow entering (fan-in) and leaving (fan-out) a procedure [8]. Harman et. al [7] demonstrate that it can also be calculated via slicing. Mapping and evaluating their approaches to Z leads to the following set of quality-based specification metrics [3]:

- *Coverage* $Cov(\psi)$: It measures the compactness of a schema by comparing the length of all possible slices to the length of the specification schema ψ.
- *Overlap* $O(\psi)$: It measures the conciseness of a schema ψ by counting those statements that are common to all of the possible slices and relates the number to the size of all slices.
- *Schema Coupling* $\chi(\Psi,\psi_i)$: It is the weighted measure of the information flow between a given schema ψ_i and all other schemas in Ψ.

3 The Study

The study is split into two parts and it aims at answering the following two questions: (a) what type of correlations exists between specification-based and code-based measures, and (b) is it possible to predict code-based measures from specification-based measures?

The Tokeneer system [5] is one of the rare, industrial-size and publicly available, formal Z specifications that comes along with a fully derived implementation. It has been developed by Praxis and the *NSA* and provides a specification for an identification station consisting of a fingerprint reader, a display and a card reader. The code, written in *ADA*, consists of $11,807$ lines of executable *ADA* code ($34,769$ lines including comments). The exceptional feature of the *ADA* files is that they contain so-called "trace unit" comments which are direct links to the corresponding sections in the formal design document, thus linking specification text (schemas) and implementation code pairs (procedures and functions) unambiguously together. The Z specification consists of $11,356$ lines of text, including $4,808$ lines of specification text. The specification itself contains $3,295$ declarations and predicates, it contains $132,088$ control dependencies and $6,145$ data-dependencies.

The subjects for this study are set of pairs of code modules (procedures and functions) and their related Z specification (schemas)[3]. However, the mapping is not always one-to-one, and it is also not

[3]The set of experimental subjects can be found in an Appendix (containing relevant background materials) at the *ViZ* [2] homepage via the link http://viz.uni-klu.ac.at/images/research/materials/fmds12-addon.pdf. Page last visited: May 2012.

total. There are a couple of trace-units that do not have a corresponding part in the implementation, and there are also links to trace-units that are non-existent. Thus, as a first step in the preparation phase of this study, a small script was written for matching the references and units automatically, sorting out spelling errors and dangling links. Then, the result of the mapping has been verified and cross-checked by hand. This process yielded 70 units with a traceable transformation of Z code to *ADA* code.

The first part of the study deals with the question of relatedness between specification-code pair measures. As we do not know whether the measures are normally distributed, three different statistical tests are used to assess the data: the Pearson's Correlation Coefficient, the Spearman's Rank Correlation Coefficient, and Kendall's Tau Correlation Coefficient. The Pearson's correlation coefficient (R_P) measures the degree of association between the variables, assuming normal distribution of the values [16, p. 212]. Though this test might not necessarily fail when the data is not normally distributed, the Pearson's test only looks for a linear correlation. It might indicate no correlation even if the data is correlated in a non-linear manner. As the data might not be normally distributed, the Spearman's rank correlation coefficient (R_S) has been chosen [16, p. 219]. It is a non-parametric test of correlation and assesses how well a monotonic function describes the association between the variables. This is done by ranking the sample data separately for each variable. Finally, the Kendall's robust correlation coefficient (R_K) is used as an alternative to the Spearman's test [6, p. 200]. It is also non-parametric and investigates the relationship among pairs of data. However, it ranks the data relatively and is able to identify partial correlations.

When a value of $\mid R \mid \in [0.8,\ 1.0]$ then it is interpreted to indicate a *strong association*. When $\mid R \mid \in [0.5,\ 0.8)$ it is interpreted to indicate a *moderate association*. When $\mid R \mid \in [0.0,\ 0.5)$ it is interpreted to indicate a *weak association* (values rounded to the third decimal place).

4 Correlation Tests

After data preparation, the study looked for linear or at least partial correlations between the sets of measures. At first, classical size-based measures are considered, and Table 1 (upper part) summarizes the results. The p-values for testing the hypothesis of no correlation against the alternative that there is a nonzero correlation are less than 0.05 for all tests. The table also shows that there is a moderate to strong relation between $CC(\psi)$ and the measures of $CL(M)$, $KNOTS(M)$ and $FOUT(M)$. The correlation values of the tests are quite similar, but there are a couple of exceptions. Compared to the Pearson test, the Spearman's rank test shows a higher correlation between most of the size-based measures and the Count Line Declarative $CLCD(M)$ measure, indicating that there might be a non-linear correlation between them. However, the Kendall's test shows weak correlation for most of the measures again. A similar situation can be observed for the measure of $FIN(M)$. Here, the Spearman test shows a slightly higher correlation than the other two tests, but it still falls into the weak association class. Interesting are the differences between the tests for the $SI(M)$ measure. The correlation to the size-based measures is not strong, but $SI(M)$ is calculated by also using the square of $FOUT(M)$, and this non-linear tendency can be seen in the slightly higher values of the Spearman tests. And yet another issue can be observed: cyclomatic complexity is (although only moderately) influenced by the number of logical OR connections in the specification. As cyclomatic complexity is related to the number of paths through the program, this observation seems also to be consistent to the use of or-combined predicates in a Z specification.

In a second step, structure-based measures have been looked at. Table 1 (lower part) summarizes the results. Again, the p-values are less than 0.05 for all tests. The correlations are not as strong as with the pure size-based measures – with one exception: the structure-based measures seem to strongly

Pearson Correlation R_P, $n = 70$, $p \leq 0.033$)									
Measure	*CL(M)*	*CLC(M)*	*CLCD(M)*	*CLCE(M)*	*CYC(M)*	*KNOTS(M)*	*FIN(M)*	*FOUT(M)*	*SI(M)*
$CC(\psi)$	0.806	0.681	0.343	0.797	0.749	0.842	0.258	0.866	0.255
$AND(\psi)$	0.538	0.507	0.369	0.519	0.586	0.477	0.366	0.482	0.384
$OR(\psi)$	0.450	0.616	0.435	0.640	0.697	0.490	0.456	0.628	0.481
Spearman's Rank Correlation R_S, $n = 70$, $p \leq 0.016$)									
$CC(\psi)$	0.784	0.742	0.653	0.797	0.770	0.783	0.418	0.849	0.615
$AND(\psi)$	0.398	0.428	0.406	0.457	0.454	0.425	0.286	0.452	0.343
$OR(\psi)$	0.697	0.731	0.699	0.760	0.748	0.704	0.497	0.774	0.619
Kendall Robust Correlation R_K, $n = 70$, $p \leq 0.025$)									
$CC(\psi)$	0.586	0.544	0.462	0.595	0.577	0.623	0.300	0.686	0.448
$AND(\psi)$	0.289	0.308	0.286	0.343	0.334	0.341	0.200	0.356	0.241
$OR(\psi)$	0.553	0.596	0.575	0.629	0.629	0.563	0.404	0.654	0.507

Pearson Correlation R_P, $n = 70$, $p \leq 0.028$)									
Measure	*CL(M)*	*CLC(M)*	*CLCD(M)*	*CLCE(M)*	*CYC(M)*	*KNOTS(M)*	*FIN(M)*	*FOUT(M)*	*SI(M)*
$v_l'(\psi)$	0.789	0.676	0.382	0.764	0.743	0.793	0.262	0.813	0.264
$v_u'(\psi)$	0.787	0.661	0.358	0.758	0.739	0.794	0.259	0.808	0.262
$DU(\psi)$	0.799	0.642	0.285	0.777	0.736	0.817	0.279	0.833	0.282
Spearman's Rank Correlation R_S, $n = 70$, $p \leq 0.002$)									
$v_l'(\psi)$	0.782	0.727	0.638	0.785	0.753	0.775	0.416	0.838	0.603
$v_u'(\psi)$	0.764	0.695	0.602	0.762	0.725	0.785	0.363	0.824	0.556
$DU(\psi)$	0.767	0.682	0.593	0.777	0.728	0.784	0.377	0.832	0.600
Kendall Robust Correlation R_K, $n = 70$, $p \leq 0.003$)									
$v_l'(\psi)$	0.603	0.543	0.460	0.602	0.583	0.628	0.305	0.691	0.441
$v_u'(\psi)$	0.565	0.502	0.431	0.552	0.533	0.619	0.262	0.655	0.402
$DU(\psi)$	0.567	0.518	0.431	0.584	0.558	0.613	0.280	0.659	0.430

Table 1: Pearson's, Spearman's and Kendall's correlation for size- and structure based Z measures.

influence the $FOUT(M)$ count. The other structure-based measures moderately to strongly influence the complexity measures $CYC(M)$ and $KNOTS(M)$. This seems to be inherent, as these measures are counting dependencies within and between the schemas. The correlation to the other *ADA*-related measures in also moderate to strong. Only the measures of $CLCD(M)$, $FIN(M)$ and $SI(M)$ do have weak correlations.

In the case of semantics-based measures the picture has to be looked at in a more differentiated way (see Table 2). At first, most of the results of the tests concerning *Coverage* are statistically not significant (higher p values are shown in bold numerals). The tests indicate no correlation between the *ADA*-based measures and *Coverage*, but the chance is high that this is wrong. In this situation scatter plots have been used to gain a better understanding of the results, but the plots confirmed the results of no correlation at all. The other tests indicate weak to moderate relations for *Overlap* and *Coupling*, but another point is interesting. *Overlap* and *Coupling* have different leading signs. This might partially be explained by the fact that overlap is an indicator for crispness. It is high when the schema is not strongly related to other parts of the specification. And coupling is higher when there are more relations to other specification schemas. An increase in the value of one measure leads to a decrease of the other measure.

To summarize, there is only weak to moderate relation between the Z-based measures and $CLCD(M)$, $FIN(M)$, and $SI(M)$. But, though not exclusively, there is some moderate to strong correlation between the Z-based and the other implementation-based measures. When just focusing, for example, on $CL(M)$, $CYC(M)$, $KNOTS(M)$, and $FOUT(M)$ and taking moderate to strong correlations into account, then the following can be observed: Firstly, they are all influenced by structure-based measures. Secondly, especially $CL(M)$, $KNOT(M)$ and $FOUT(M)$ do have strong correlations to the Z measures. The next section now uses the Z measures to provide regression formulas for the most suitable *ADA*-based measures.

Semantics-based Correlation, $n = 70$										
		Pearson			Spearman			Kendall		
		$Cov(\psi)$	$O(\psi)$	$\chi(\psi)$	$Cov(\psi)$	$O(\psi)$	$\chi(\psi)$	$Cov(\psi)$	$O(\psi)$	$\chi(\psi)$
CL(M)	R	0.104	-.623	0.646	0.054	-.616	0.686	0.042	-.495	0.480
	p	**0.391**	0.000	0.000	**0.660**	0.000	0.000	**0.624**	0.000	0.000
CLC(M)	R	0.184	-.466	0.414	0.186	-.480	0.541	0.146	-.364	0.379
	p	**0.127**	0.000	0.000	**0.124**	0.000	0.000	**0.085**	0.000	0.000
CLCD(M)	R	0.229	-.215	0.116	0.243	-.369	0.433	0.190	-.280	0.310
	p	**0.057**	**0.075**	**0.340**	0.042	0.002	0.000	0.026	0.003	0.000
CLCE(M)	R	0.126	-.559	0.546	0.110	-.587	0.636	0.079	-.461	0.440
	p	**0.300**	0.000	0.000	**0.363**	0.000	0.000	**0.355**	0.000	0.000
CYC(M)	R	0.148	-.534	0.509	0.137	-.531	0.590	0.103	-.416	0.412
	p	**0.221**	0.000	0.000	**0.258**	0.000	0.000	**0.234**	0.000	0.000
KNOTS(M)	R	0.062	-.587	0.637	-.018	-.629	0.721	-.034	-.530	0.542
	p	**0.613**	0.000	0.000	**0.884**	0.000	0.000	**0.708**	0.000	0.000
FIN(M)	R	0.150	-.224	0.212	0.245	-.170	0.267	0.184	-.132	0.193
	p	**0.216**	**0.062**	**0.078**	0.041	**0.160**	0.026	0.034	**0.164**	0.026
FOUT(M)	R	0.096	-.572	0.582	0.046	-.599	0.722	-.008	-.479	0.541
	p	**0.430**	0.000	0.000	**0.704**	0.000	0.000	**0.930**	0.000	0.000
SI(M)	R	0.123	-.235	0.227	0.220	-.406	0.480	0.159	-.315	0.339
	p	**0.309**	0.050	**0.059**	**0.067**	0.000	0.000	**0.063**	0.001	0.000

Table 2: Pearson, Spearman and Kendall for semantics-based measures.

5 Prediction Models

According to a rule of thumb in regression [10, p.3], the appropriate number of independent variables for a prediction is not more than one fifth of the sample size. Thus, the eleven Z measures presented in Section 2 can be considered to be sufficient and they are all selected to form the maximum model for 70 observations in this study. Among several systematic methods for restricting the maximum model, a backward elimination procedure [10, p.8] with a threshold of 0.4 for the P-values is selected. This means that a maximum regression model with all eleven independent variables is built. Then all the variables with a P-value of more than 0.4 are eliminated. Then, again, another regression model with the reduced number of variables is built, iterating until there is no variable with a P-value higher than 0.4.

Table 3 summarizes the final result of this procedure for the five remaining code metrics (as measures with a P-Value higher than 0.4 have been eliminated). The table, for example, shows that for the calculation of the cyclomatic complexity $CYC(M)$ of an *ADA* module, $CC(\psi)$, $Cov(\psi)$ and $OR(\psi)$ are best for being used in the regression formula. The level of confidence can be explained by the values of *Significance-F*. If the level of acceptable confidence should be 95% and higher, then all the code metrics with F-values of less than 0.05 can be considered predictable using the metrics in Z. All the values for F in Table 3 show that there is a high reliability on the results of the regressions. The value of *Adjusted R-Square* can be interpreted as an indicator for the precision level of the prediction. In our case the values are between 0.620 and 0.840, indicating that the regression models are relatively precise for $FOUT(M)$, $KNOTS(M)$ and $CL(M)$ and even more precise for $CLE(M)$ and $CYC(M)$. With these values at hand it makes sense to predict code metrics, and the resulting formulas are as follows:

$$CL(M) = 3.099CC(\psi) - 1.237USE(\psi) + 2.557AND(\psi) - 41.735OR(\psi) - 9.873$$
$$CLCE(M) = 0.516CC(\psi) - 0.003v'_u(\psi) - 0.477DEF(\psi) + 5.458OR(\psi) + 5.819$$
$$CYC(M) = 0.015CC(\psi) + 4.349Cov(\psi) - 2.107O(\psi) + 1.082OR(\psi) + 1.666$$
$$KNOTS(M) = 0.121CC(\psi) - 0.001v'_u(\psi) - 0.017USE(\psi) - 0.092DEF(\psi) + 0.027AND(\psi) - 0.882$$
$$FOUT(M) = 0.198CC(\psi) - 0.107v'_l(\psi) - 0.001v'_u(\psi) - 0.211DEF(\psi) + 1.220OR(\psi) + 0.344$$

Results of the backward elimination procedure (values $P \leq 0.4$)						
Paramter		*CL(M)*	*CLE(M)*	*CYC(M)*	*KNOTS(M)*	*FOUT(M)*
Adjusted R-Square		0.720	0.680	0.620	0.760	0.840
Significance F		5E-18	2E-16	5E-14	7E-20	1.5E-25
$P-Value$	$CC(\psi)$	0.001	4E-4	0.003	1E-6	3E-11
	$v'_l(\psi)$	——	—	—	—	0.270
	$v'_u(\psi)$	——	0.005	—	0.004	0.000
	$DU(\psi)$	——	—	—	—	—
	$O(\psi)$	——	—	—	—	—
	$Cov(\psi)$	——	—	0.280	—	—
	$\chi(\psi)$	——	—	—	—	—
	$AND(\psi)$	7E-5	—	—	0.030	—
	$OR(\psi)$	5E-4	0.001	4E-4	—	6E-5
	$DEF(\psi)$	——	0.070	—	0.020	1E-5
	$USE(\psi)$	0.034	—	—	0.320	—

Table 3: Adjusted R-Square, Significance F and P-Values after applying the backward elimination procedure for maximum model identification. P-Values higher than 0.4 are represented by dashes.

6 Threats to Validity

With the results of the study the question of validity arises. Considering internal validity, single group and multiple group threats, as well as social threats cannot arise. The only threat that might have an impact on the outcome of the study is the software used to generate and calculate the measures. The software components involved are the *CZT* parser [11], the slicing environment *ViZ* [2], *Matlab R*2007*b*, *Microsoft Excel* 2010 and *SPSS* 14:0. *Excel* is a standard spreadsheet application. *Matlab* and *SPSS* are numerical computer environments used for the statistical analysis. Both tools have been used alternately to verify the results of the analysis. It is very unlikely that the data from both environments is erroneous. The *CZT* parser is being developed as a *SourceForge* project since 2003 and it is available in a stable release. The slicing environment *ViZ* has been developed in the year 2003 and it is also part of a couple of extensions which led to systematic validations during development.

Concerning the selection validity, the publicly available schemas and *ADA* procedures and functions have been chosen with care, following the links provided by the developers. It is important to note that the specification used in this study had to be modified a bit in order to be accepted by the *CZT* parser. This meant to introduce some hard spaces and, eventually, also to replace the "$\widehat{=}$" symbol by the "$==$" sign. In order to rule out the possibility of coincidental changes of line breaks or identifier names, both files were again compared afterwards, using a professional file-compare software.

7 Conclusion

In this study, consisting of 70 experimental subjects, the feasibility of confidently predicting software measures based on formal specifications has been demonstrated. The correlations found between the different size-, structure-, and semantics-based measures and the implementation metrics promise of being able to predict size and complexity attributes as well as enables to estimate likely costs and efforts.

The study describes only the first link in the chain of associations between the documents created during software development, but it confirms the observations of Samson et.al. [17] who conducted a similar study (with 9 experimental subjects) several years ago. Specification-based measures are not difficult to calculate, thus they can and also *should* be collected at the beginning of a project. The results of the study indicate that it pays off.

References

[1] Andreas Bollin (2004): *Specification Comprehension – Reducing the Complexity of Specifications*. Ph.D. thesis, University of Klagenfurt.

[2] Andreas Bollin (2007): *Concept Location in Formal Specifications*. Journal of Software Maintenance and Evolution: Research and Practice Manuscript submitted Jan. 2007, doi:10.1002/smr.363.

[3] Andreas Bollin (2010): *Slice-based Formal Specifiation Measures – Mapping Coupling and Cohesion Measures to Formal Z*. In Cèsar Muñoz, editor: Proceedings of the Second NASA Formal Methods Symposium, NASA/CP-2010-216215, NASA, Langley Research Center, pp. 24–34.

[4] Juei Chang & Debra J. Richardson (1994): *Static and Dynamic Specification Slicing*. Technical Report, Department of Information and Computer Science, University of California.

[5] Rod Chapman (2009): *The Tokeneer ID Station – Overview and Readers Guide. S.P1229.81.8. Issue: 1.4*. Praxis High Integrity Systems.

[6] Norman E. Fenton & Shari Lawrence Pfleeger (1989): *Software Metrics*, 2nd edition. Thompson Press.

[7] Mark Harman, Margaret Okulawon, Bala Sivagurunathan & Sebastian Danicic (1997): *Slice-based measurement of coupling*. In: Proceedings of the ICSE workshop on Process Modelling and Empirical Studies of Software Evolution. Boston, Massachusetts, IEEE Computer Society, Los Alamitos, CA, USA, pp. 28–32.

[8] Sally M. Henry & Dennis G. Kafura (1981): *Software structure metrics based on information flow*. IEEE Transactions on Software Engineering 7(5), pp. 510–518, doi:10.1109/TSE.1981.231113.

[9] Magne Jørgensen (2004): *A review of studies on expert estimation of software development effort*. Journal of Systems and Software 70(1–2), pp. 37–60, doi:10.1016/S0164-1212(02)00156-5.

[10] Pia Veldt Larsen (2008): *ST111: Regression and analysis of variance – Module 8: Selecting regression models*. Manual from statmaster.sdu.dk/courses/st111/module08, Syddansk University, Department of Statistics.

[11] Petra Malik (2011): *A retrospective on CZT*. Software – Practice and Experience 41(2), pp. 179–188, doi:10.1002/spe.1015.

[12] Roland T. Mittermeir & Andreas Bollin (2003): *Demand-driven Specification Partitioning*. In: Proceedings of the 5th Joint Modular Languages Conference, JMLC'03, pp. 241–253, doi:10.1007/978-3-540-45213-3_30.

[13] Tomohiro Oda & Keijiri Araki (1993): *Specification slicing in a formal methods software development*. In: Seventeenth Annual International Computer Software and Applications Conference, IEEE Computer Socienty Press, pp. 313–319, doi:10.1109/CMPSAC.1993.404234.

[14] Linda M. Ott & Jeffrey J. Thus (1989): *The Relationship between Slices and Module Cohesion*. In: 11th International Conference on Software Engineering, IEEE Computer Society, Los Alamitos, CA, USA, pp. 198–204, doi:10.1145/74587.74614.

[15] Lawrence H. Putnam & Ware Myers (2003): *Five Core Metrics: The Intelligence Behind Successful Software Management*. Dorset House.

[16] D. G. Rees (2003): *Essential Statistics*, 4th edition. Chapman & Hall.

[17] W.B. Samson, Denis G. Nevill & P.I. Dugard (1987): *Predictive software metrics based on a formal specification*. In: Information and Software Technology, *5* 29, pp. 242–248.

[18] Martin J. Shepperd & Darrel C. Ince (1990): *The use of metrics in the early detection of design errors*. In: Proceedings of the European Software Engineering Conference 90, pp. 67–85.

[19] Harry M. Sneed, Richard Seidl & Manfred Baumgartner (2010): *Software in Zahlen*. Carl Hanser Verlag.

[20] Abdollah Tabareh (2011): *Predictive Software Measures Based on Formal Z Specifications*. Master's thesis, University of Gothenburg - Department of Computer Science and Engineering.

[21] Mark Weiser (1982): *Program slicing*. In: Proceedings of the 5^{th} International Conference on Software Engineering, IEEE Press, Piscataway, NJ, USA, pp. 439–449.