Slice-based Formal Specification Measures – Mapping Coupling and Cohesion Measures to Formal Z

Andreas Bollin
Alpen-Adria Universität Klagenfurt
Klagenfurt, Austria
Andreas.Bollin@uni-klu.ac.at

Abstract

This paper demonstrates that existing slice-based measures can reasonably be mapped to the field of state-based specification languages. By making use of Z specifications this contribution renews the idea of slice-profiles and derives coupling and cohesion measures for them. The measures are then assessed by taking a critical look at their sensitiveness in respect to modifications on the specification source. The presented study shows that slice-based coupling and cohesion measures have the potential to be used as quality indicators for specifications as they reflect the changes in the structure of a specification as accustomed from their program-related pendants.

1 Introduction

In one of the rare articles concerning the relation between specifications and code, Samson, Nevill, and Dugard [13] demonstrate a strong quantitative correlation between size-based specification metrics and the related pendant of software code. Their assumption is that a meaningful set of (complexity and quality) measures could help in estimating product measures and development effort at a much earlier stage. Complexity can be described by size-based attributes, but is it reasonable to measure the quality of a specification? This contribution takes a closer look at this problem.

Quality considerations are sophisticated. Besides the question of what a ”good specification” looks like, quality-based measures (as in use with program code) are not so easily transformed to specifications. One reason is that such measures are usually based on control/data dependency considerations – concepts that are either not at all or only implicitly available. However, various authors demonstrated in [11, 4, 10, 17] that a reconstruction of the necessary dependencies ameliorates the situation and enables continuative techniques like slicing and chunking. And with that, slice-based measures (which are often taken as the basis for quality considerations) can be mapped to formal specifications, too. What would be the benefits of such measures?

As presumed by Samson et. al., with experiences from a large collection of specifications and implementations at hand, product and development estimates could be calculated at much earlier stages. But there is another benefit. When the measures are sensitive and react instantly to changes in the specifications, considerations, e.g. concerning deterioration effects, could be made, too.

The main objective of this contribution is now to investigate whether slice-based quality measures can reasonably be transformed to formal specifications. It does not invent new measures, but it maps the ideas behind the definitions of coupling and cohesion measures to the world of formal specification languages. Additionally, it looks for possible limitations. Based on Z [14], the mapping is described in some details and the outcome is assessed in respect to its expressiveness and sensitiveness.

This paper is structured as follows: Section 2 introduces specification slices and takes them as the basis for the slice-based measures mentioned above. Section 3 discusses the effects on the measures by making use of sample specifications, and Section 4 concludes the work with a short outlook.

2 Background

The motivation behind analyzing slice-based coupling and cohesion measures goes back to a paper of Meyers and Binkley [8]. In their empirical study they take a closer look at these measures and demonstrate that the values of coupling and cohesion can also be used for assessing deterioration effects. As formal specifications evolve, too, it would be interesting to see whether changes in the specification code show a similar behavior of these measures. As a necessary first step, a reasonable transformation of the original definitions of the measures to the world of formal specifications has to be found. This section demonstrates how this can be done for Z [14].

2.1 Slice-based Coupling and Cohesion

Coupling is a measure for the strength of inter-component connections, and cohesion is a measure for the mutual affinity of sub-components of a component. Within the range of this contribution we are interested in *how* these measures are calculated and *what* they indicate. As adumbrated in the introduction, a practical way in calculating coupling and cohesion measures is to make use of slices.

Weiser [15, 16] introduced five slice-based measures for cohesion: *Tightness*, *Coverage*, *Overlap*, *Parallelism*, and *Clustering*. Ott and Thuss [12] partly formalized these measures, and this contribution makes use of their formalization. *Coupling* was originally defined as the number of local information flow entering (fan-in) and leaving (fan-out) a procedure [7]. Harman et. al demonstrate in [6] that it can also be calculated via slicing. Furthermore, they show that the use of slices not only enables the detection of coupling, it can also be used to determine the ”bandwidth” of the existing information flow. Their notion of information flow is also used in this contribution.

2.2 Specification Slices and Slice Profiles

For the calculation of coupling and cohesion measures, sets of slices and their intersections (comparable to the use of slice profiles in [12]) are needed. For state-based specifications the technique of slicing was introduced by Oda and Araki [11], informally redefined by Chang and Richardson [4], and then refined by Bollin [1]. His Java prototype has been extended in the recent years. It now supports slicing, chunking, and concept location of Z specifications [3]. The technical details of the identification of dependencies are not relevant within the scope of this paper, but the basic idea is quite simple:

First, the specification is dismantled into its basic elements called primes[1] by making use of the *CZT* parser [9]. The primes are mapped to a graph called *SRN* (for *Specification Relationship Net*). Then, by following the approach of Chang and Richardson and Bollin [4, 1] control and data dependencies are reconstructed (via a syntactical approximation to the semantical analysis). The *SRN* gets annotated by this dependency information, yielding an *Augmented SRN* (*ASRN* for short).

The *ASRN* serves the same purpose as the system dependence graph used by the approaches described in [8, p.4]. Based on this data structure, slicing works as follows: a set of vertices (representing the point of interest) in the *ASRN* is taken as starting point, and, by following the dependencies existing in the graph, further primes are aggregated, resulting in the designated specification slice. The transformation between a specification Ψ and its *ASRN* is defined in a bijective manner. So, when talking about a specification it can be either the textual representation (consisting of a set of primes) or its *ASRN* representation (consisting of vertices representing the primes).

[1] Basically, primes are the predicates of the specification and are later represented as vertices in an augmented graph. When they represent predicates of the precondition of a schema they are called precondition primes, and when they form predicates that represent after-states they are called postcondition primes.

Harman et. al [6] and Ott and Thuss [12] use different types of slices for their calculation of coupling and cohesion values. This situation is dealt with hereinafter by generating two variants of the static specification slices: for *coupling* the slices are calculated by following the dependencies in a transitive backward manner, for the values of *cohesion* the slices are calculated by combining the dependencies in a forward and backward manner. Specification slices and slice profiles (the collection of slices for a specific schema operation) are then defined as follows:

Definition 1. Static Specification Slice. *Let Ψ be a formal Z specification, ψ one schema out of Ψ, and V a set of primes v out of ψ. $SSlice_{fb}(\psi, V)$ is called static forward/backward specification slice of ψ for primes V. It is calculated by generating a transitive forward and backward slice with V as the starting point of interest. When the slice is generated in a transitive and backward manner, it is called static backward slice $SSlice_b(\psi, V)$.*

Definition 2. Slice Profile, Slice Intersection, Slice Union. *Let Ψ be a formal Z specification, ψ one schema out of Ψ, and V the set of primes v representing all postcondition primes in ψ. The set of all possible static specification slices ($SSlice_{fb}(\psi, \{v\})$ or $SSlice_b(\psi, \{v\})$, with $v \in V$) is called Slice Profile ($SP(\psi)$). The intersection of the slices in $SP(\psi)$ is called Slice Intersection ($SP_{int}(\psi)$). The union of all slices in $SP(\psi)$ is called Slice Union ($SU(\psi)$).*

2.3 Cohesion

With the introduction of slice profiles it is possible to provide the definitions of cohesion measures (as introduced in the work of Ott and Thuss [12]). The values for cohesion are calculated only for a given schema. As slices and slice profiles might contain primes from other schemata (due to inter-schema dependencies), the following definitions restrict the set of primes in the slice profile to the schema.

Definition 3. Tightness. *Let Ψ be a formal Z specification, ψ one schema out of Ψ, $SP(\psi)$ its slice profile, and $SP_{int}(\psi)$ its slice intersection. Then Tightness $\tau(\psi)$ is the ratio of the size of the slice intersection to the size of ψ. It is defined as follows:*

$$\tau(\psi) = \frac{\mid SP_{int}(\psi) \cap \psi \mid}{\mid \psi \mid}$$

Definition 4. MinCoverage, Coverage, MaxCoverage. *Let Ψ be a formal Z specification, ψ one schema out of Ψ, and $SP(\psi)$ its slice profile containing n slices. MinCoverage $Cov_{min}(\psi)$ expresses the ratio between the smallest slice SP_{i-min} in $SP(\psi)$ and the number of predicate vertices in ψ. Coverage $Cov(\psi)$ relates the sizes of all possible specification slices SP_i ($SP_i \in SP(\psi)$) to the size of ψ. MaxCoverage $Cov_{max}(\psi)$ expresses the ratio of the largest slice SP_{i-max} in the slice profile $SP(\psi)$ and the number of vertices in ψ. They are defined as follows:*

$$Cov_{min}(\psi) = \frac{1}{\mid \psi \mid} \mid SP_{i-min} \cap \psi \mid \qquad Cov(\psi) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mid SP_i \cap \psi \mid}{\mid \psi \mid} \qquad Cov_{max}(\psi) = \frac{1}{\mid \psi \mid} \mid SP_{i-max} \cap \psi \mid$$

Definition 5. Overlap. *Let Ψ be a formal Z specification, ψ one schema out of Ψ, $SP(\psi)$ its slice profile containing n slices, and SP_{int} its slice intersection. Then Overlap $O(\psi)$ measures how many primes are common to all possible specification slices SP_i ($SP_i \in SP(\psi)$). It is defined as follows:*

$$O(\psi) = \frac{1}{n} \sum_{i=1}^{n} \frac{\mid SP_{int} \cap \psi \mid}{\mid SP_i \cap \psi \mid}$$

Tightness measures the number of primes that are common to every slice. The definition is based on the size[2] of the slice intersection. Coverage is split into three different measures: *Minimum Coverage* looks at the size of the smallest slice and relates it to the size of the specification, *Coverage* looks at the size of the slices, but it takes all slices and compares them to the size of the specification, and *Maximum Coverage* looks at the size of the largest slice and relates it to the size of the specification. Finally, *Overlap* looks at the slice intersection and determines how many primes are common to all slices.

[2]Please note that within the context of all definitions *size* counts the number of primes in the *ASRN*.

2.4 Coupling

The calculation of coupling follows the definitions to be found in [6]. First, Inter-Schema Flow F is specified. It describes how many primes of the slices in the slice union are outside of the schema. Inter-Schema Coupling then computes the normalized ratio of this flow in both directions.

Definition 6. Inter-schema Flow and Coupling. *Let Ψ be a formal Z specification and ψ_s and ψ_d two schemata out of Ψ. Inter-Schema Flow between the two schemata $F(\psi_s, \psi_d)$ is the ratio of the primes of $SU(\psi_d)$ that are in ψ_s and that of the size of ψ_s. Inter-Schema Coupling between the two schemata $C(\psi_s, \psi_d)$ measures the Inter-Schema Flow in both directions. They are defined as follows:*

$$F(\psi_s, \psi_d) = \frac{\mid SU(\psi_d) \cap \psi_s \mid}{\mid \psi_s \mid} \qquad C(\psi_s, \psi_d) = \frac{F(\psi_s, \psi_d) \times \mid \psi_s \mid \; + F(\psi_d, \psi_s) \times \mid \psi_d \mid}{\mid \psi_s \mid + \mid \psi_d \mid}$$

Schema coupling is calculated by considering the Inter-Schema Coupling values to all other schemata.

Definition 7. Schema Coupling. *Let Ψ be a formal Z specification and ψ_i one schema in Ψ. Then Schema Coupling $\chi(\psi_i)$ is the weighted measures of the Inter-Schema Coupling of ψ_i to all other n schemata ψ_j in $\Psi \setminus \psi_i$. It is defined as follows:*

$$\chi(\psi_i) = \frac{\sum_{j=1}^{n} C(\psi_i, \psi_j) \times \mid \psi_j \mid}{\sum_{j=1}^{n} \mid \psi_j \mid}$$

With the measures in this section it is possible to assign attributes to a formal Z specification. However, with the mapping a connection to quality has been so far not empirically justified. On the other hand, the slice-based measures have carefully been transformed to Z. There is a chance that, when observing *changes* of these values for a given specification, one might defer useful properties.

3 Sensitivity of Slice-based Measures

By following the strategy that Thuss and Ott [12] used for their validations, we now investigate the *sensitivity* of the measures with respect to *representative changes* of the specifications. The advantage is that for such a study only small-sized sample specifications are necessary to explore the effects.

3.1 Sensitivity of Cohesion

The first objective is to determine whether the transformed measures for cohesion are sensitive to changes in the internal structure of the specification. The following operations are considered:

- O1 Adding a precondition-prime. This means that this prime ”controls” the evaluation of the other primes in the schema. With it, the internal semantic connections are extended. Mapped to a potential implementation, this could mean that an if-clause is added to the code, enveloping all other statements in the method. This operation should slightly increase coverage.
- O2 Adding a prime that specifies an after-state and that is not related to all the other predicates in the schema. In this case the predicate introduces new ”trains of thought”. Mapped to a subsequent implementation, this could mean that a new output- or state-relevant statement (not or only fractionally related to the other statements) is added. With it, a new slice is added to the slice-profile. The slice intersection is very likely smaller than before, thus reducing the values for cohesion.
- O3 Adding a prime that specifies an after-state and that is furthermore related to all other primes in the schema. In this case the predicate extends existing ”trains of thought” (as there are references to all existing ones). Mapped to a possible implementation, it is very likely that a new output- or state-relevant statement, related to all other statements, is added. If at all, this increases the set of intersection slices. And with that, it also raises the values of coupling.

$SP(\psi)$			Specifications	Measures
		—	*Test*1 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $n' = n + 1$	$\#SP_{int}(\psi) = 1$ $\tau(\psi) = 1.00$ $Cov_{min}(\psi) = 1.00$ $Cov(\psi) = 1.00$ $Cov_{max}(\psi) = 1.00$ $O(\psi) = 1.00$
	—	—	*Test*2 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $n' = n + 1$ $m' = m + 1$	$\#SP_{int}(\psi) = 0$ $\tau(\psi) = 0.00$ $Cov_{min}(\psi) = 0.50$ $Cov(\psi) = 0.50$ $Cov_{max}(\psi) = 0.50$ $O(\psi) = 0.00$
	— —	— —	*Test*3 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $delta? : \mathbb{N}$ $delta? > 0$ $n' = n + delta?$ $m' = m - delta?$	$\#SP_{int}(\psi) = 1$ $\tau(\psi) = 0.33$ $Cov_{min}(\psi) = 0.67$ $Cov(\psi) = 0.67$ $Cov_{max}(\psi) = 0.67$ $O(\psi) = 0.50$
	— — —	— —	*Test*4 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $delta? : \mathbb{N}$ $set? : \mathbb{P}\mathbb{N}$ $delta? > 0$ $set? \neq \varnothing$ $n' = n + delta?$ $m' = m - delta?$	$\#SP_{int}(\psi) = 2$ $\tau(\psi) = 0.50$ $Cov_{min}(\psi) = 0.75$ $Cov(\psi) = 0.75$ $Cov_{max}(\psi) = 0.75$ $O(\psi) = 0.67$

$SP(\psi)$			Specifications	Measures
— — —	— — — 	— — —	*Test*5 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $delta? : \mathbb{N}$ $set? : \mathbb{P}\mathbb{N}$ $p, p' : \mathbb{N}$ $delta? > 0$ $set? \neq \varnothing$ $n' = n + delta?$ $m' = m - delta?$ $p' = p + delta?$	$\#SP_{int}(\psi) = 2$ $\tau(\psi) = 0.40$ $Cov_{min}(\psi) = 0.60$ $Cov(\psi) = 0.60$ $Cov_{max}(\psi) = 0.60$ $O(\psi) = 0.33$
— — — —	— — — 	— — — 	*Test*6 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $delta? : \mathbb{N}$ $set? : \mathbb{P}\mathbb{N}$ $p, p' : \mathbb{N}$ $delta? > 0$ $set? \neq \varnothing$ $n' = n + delta?$ $m' = m - delta?$ $p' = p + n$	$\#SP_{int}(\psi) = 2$ $\tau(\psi) = 0.40$ $Cov_{min}(\psi) = 0.60$ $Cov(\psi) = 0.73$ $Cov_{max}(\psi) = 0.80$ $O(\psi) = 0.56$
    	 — — —	 — — —	*Test*7 $n, n' : \mathbb{N}$ $m, m' : \mathbb{N}$ $delta? : \mathbb{N}$ $set? : \mathbb{P}\mathbb{N}$ $delta? > 0$ $set? \neq \varnothing$ $n' = n + delta?$ $m' = m - n$	$\#SP_{int}(\psi) = 4$ $\tau(\psi) = 1.00$ $Cov_{min}(\psi) = 1.00$ $Cov(\psi) = 1.00$ $Cov_{max}(\psi) = 1.00$ $O(\psi) = 1.00$

Figure 1: Z specifications of raising sizes. On the left side of the table the slices (and thus the slice-profile) are visualized, on the right side the values for cohesion are presented.

Based on the assumption that schema operations are often mapped to methods (or procedures) as described in operations $O1$ to $O3$, the following working hypothesis can be posted:

Hypothesis 1. *A structural change of type $O1$, $O2$ or $O3$ in a schema operation influences the values for cohesion. Adding a predicate prime to the schema according to operations $O1$ or $O3$ increases the values (or leaves them unchanged), adding a prime according to operation $O2$ decreases the values (or leaves them unchanged). Reversing the operations also reverses the effect on the measures.*

There are situations where, due to a large number of dependencies, a method or a schema operation already has reached the maximum values for cohesion. These special cases are the reason why the values might also be unchanged (and Sec. 3.3 reconsiders this issue in more details).

Hypothesis 1 is now checked by using small sample schema operations (called *Test*1 to *Test*7 in Fig. 1). At first let us start with a simple Z schema operation called *Test*1. It contains a prime that increases the value of n by one. As there is only one slice, the slice intersection only contains one element. The values of cohesion are all 1. Then another prime ($m' = m + 1$, prescribing an after-state) is

added to the schema (which is an operation of class $O2$), yielding operation *Test*2. With this new prime a new "functionality" has been introduced to the schema. The values for cohesion are reduced as the slice intersection is empty. Tightness and Overlap are zero, the rest of the values are equal to $\frac{1}{2}$. Then, in *Test*3, the prime $delta? > 0$ is added to the schema. This prime is a precondition prime and thus the operation belongs to class $O1$. With this, all the values of cohesion increase. As the slice intersection contains one prime only ($delta? > 0$), its size is $\#SP_{int}(\psi) = 1$. With that, the values for cohesion result in: $\tau = \frac{1}{3}$, $Cov_{min} = \frac{1}{3} \times 2$, $Cov = \frac{1}{2} \times (\frac{2}{3} + \frac{2}{3})$, $Cov_{max} = \frac{1}{3} \times 2$, and $O = \frac{1}{2} \times (\frac{1}{2} + \frac{1}{2})$.

*Test*4 adds another precondition prime $set? \neq \varnothing$ to the schema (operation $O1$). This yields an increase in the values of cohesion. The reason is the increase in size of the slice intersection. *Test*5 adds another prime containing an after state identifier to the schema operation. The prime $p' = p + delta?$ is not related to the other primes prescribing after-states, so this change is an operation of class $O3$. The size of the slice intersection stays the same, only the size of the schema increases. As a result the values for cohesion decrease.

Now let us take a look at situations when predicates that are partly related to existing predicates are added to the schema. *Test*6 is an extension of *Test*4, but this time the new prime $p' = p + n$ uses the identifier n which is also defined by the postcondition prime $n' = n + delta?$. On the other hand it does not refer to the third postcondition prime $m' = m - delta?$, and so the operation belongs to class $O2$. The values for cohesion consequently decrease. On contrary, *Test*7 is a modification of *Test*3. The prime $m' = m - n$ is added, and so it is related to all other postcondition primes in the schema. With this operation the values for cohesion increase, again.

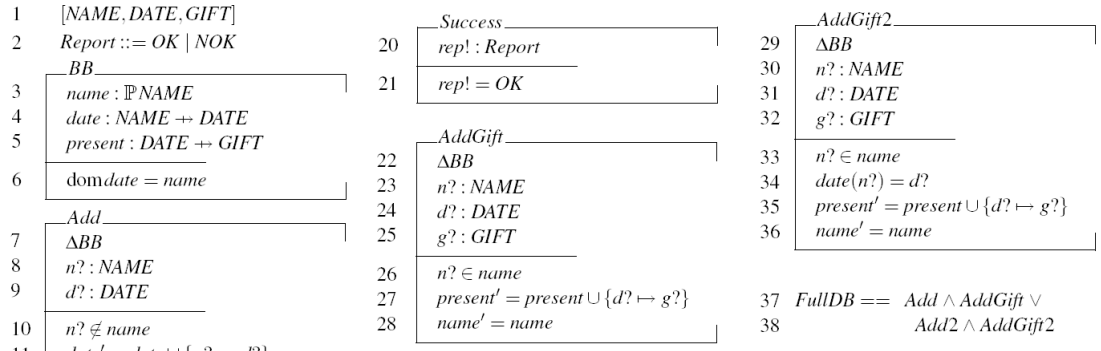
Due to reasons of space the example schemata above only contain simple predicates. But they are sufficient to demonstrate the influence of structural changes. In Z there are several schema operators that complicate the situations, but by further analyzing the formulas of the measures one observes the following behavior:

- Cohesion will increase when (a) at least one postcondition exists and a precondition prime is added, (b) a postcondition prime that is related to some, but not to all, other postcondition primes in the schema is added, (c) a postcondition prime that is not related to the other postcondition primes is removed.
- Cohesion stays the same when (a) a postcondition prime that is related to all other existing postcondition primes is added or removed and the other existing primes are already fully inter-related, (b) a pre- or postcondition prime is added and there is no postcondition prime.
- Cohesion will decrease when (a) a postcondition prime that is not related to the other postcondition primes is added, (b) a precondition prime is removed and there is at least one postcondition prime, (c) a postcondition prime that is related to the other postcondition primes is removed.

The values for a derived implementation would very likely react to these changes in the same way, so the sample specifications and the analysis of the formulas seems to confirm our working hypothesis. The measures are sensitive to changes in the underlying specification, and the observed changes are in such a way that an increase in internal connectivity also leads to an increase of the values of the measures. Conversely, a decrease in connectivity also leads to a decrease of the values of cohesion.

3.2 Sensitivity of Coupling

The next step is the evaluation of coupling. As mentioned above, specification coupling measures the mutual interdependence between different schemata. According to Harman et. al [6] it is an advantage to use slice-based measures as they measure the "bandwidth" of the information flow. In our case, this



Schema	Coupling	τ	Cov.	O
(*S*1) *BB*	0.775	1.000	1.000	1.000
(*S*2) *Add*	**0.724**	0.750	0.875	0.958
(*S*3) *Add*2	**0.737**	0.800	0.867	0.972
(*S*4) *Success*	0.000	1.000	1.000	1.000
(*S*5) *AddGift*	**0.724**	0.750	0.875	0.958
(*S*6) *AddGift*2	**0.737**	0.800	0.900	0.958

Figure 2: Z example for analyzing the effect of slice-based coupling and values for coupling and cohesion for the six Z schemata (omitting the *FullDB* operation schema).

flow is defined as the relative amount of the size of a set of slices of one schema that lies inside another schema. This flow depends upon control- and data-relationships, so an increase in the number of these dependencies should increase the value of coupling. Reducing the number of relations should decrease the value of coupling. There are no differences between the definitions of the measure, be it for traditional programs or be it for formal specifications.

The next hypothesis focuses on the sensitiveness to structural changes within a formal Z specification. Especially, an increase (or decrease) in inter-schema relations should be reflected correctly[3].

Hypothesis 2. *A change in the number of relations between two schemata in a formal specification generally leads to a change in the value of coupling. Adding a predicate prime to one schema that introduces new dependencies to primes in the other schema increases the value of coupling (or leaves it unchanged). Reversing the change also reverses the effect on the measure.*

For our considerations we now make use of a small specification (see Fig. 2) which is an extension (and intentionally unfinished version) of the birthday-book specification out of Spivey [14, pp.3–6].

The specification consists of one state space called *BB* (for *Birthday Book*) which stores the names of friends, their birthday dates, *and* a small gift. Consequently, there are two operations (*Add* and *AddGift*) for adding them to the database. The operations are not total, but are sufficient for our examinations. In order to analyze the effect of added pre- and postcondition primes, these operations are available in two versions. Finally, there is an operation called *Success* which should (later on) indicate the success of an operation. However, at the moment this operation is not related to any of the other operations.

The values of schema coupling are summarized in Fig. 3. As expected, the *Success* operation has a value of coupling equal to zero. There are no connections to the state space *BB* and also no connections

[3] Again, there are situations where, due to a high number of dependencies, the value of coupling might stay unchanged.

F	$(S1)$	$(S2)$	$(S3)$	$(S4)$	$(S5)$	$(S6)$
$(S1)$	1.000	**1.000**	1.000	0.000	1.000	1.000
$(S2)$	**0.750**	1.000	0.750	0.000	0.750	0.750
$(S3)$	0.800	0.800	1.000	0.000	0.800	0.800
$(S4)$	0.000	0.000	0.000	1.000	0.000	0.000
$(S5)$	0.750	0.750	0.750	0.000	1.000	0.750
$(S6)$	0.800	0.800	0.800	0.000	0.800	1.000

C	$(S1)$	$(S2)$	$(S3)$	$(S4)$	$(S5)$	$(S6)$
$(S1)$	1.000	**0.800**	**0.833**	0.000	0.800	0.833
$(S2)$	0.800	1.000	0.778	0.000	**0.750**	**0.778**
$(S3)$	0.833	0.778	1.000	0.000	0.778	0.800
$(S4)$	0.000	0.000	0.000	1.000	0.000	0.000
$(S5)$	**0.800**	0.750	0.778	0.000	1.000	0.778
$(S6)$	**0.833**	0.778	0.818	0.000	**0.778**	1.000

Figure 3: Values for Inter-Schema Flow $F(\psi_1,\psi_2)$ and Inter-Schema Coupling $C(\psi_1,\psi_2)$. The abbreviations $S1$ to $S6$ refer to the schema names mentioned in Fig. 2.

to the other operation schemata. On the other hand, the values for the other operations differ (though their syntactical composition is more or less equivalent). With $n = 6$ operations there are 15 different combinations and thus possibly 15 values for Inter-Schema Coupling. Within the scope of this contribution we will focus on three combinations that are of most interest in this schema constellation.

As a first example we look at the operations *Add* and *Add*2. The difference between them is made up by just one prime, namely $name' = name \cup \{n?\}$ at line 18. In fact, in the context of the specification this postcondition prime is redundant (as the state invariant at line 6 would guarantee that the added name is also in the set of names). But syntactically this prime is sufficient to increase the set of dependencies. Both operations include the state-space, which means that there is a relation between the postcondition primes and the invariant. This introduces a new "flow of control", which increases the bandwidth and thus the value for coupling (from 0.724 to 0.737 in Fig. 2).

Fig. 3 presents the values for Inter-Schema Flow and Coupling, and they make this difference more visible. Inter-Schema Flow $F(BB,Add)$ is $\frac{|SU(Add)\cap BB|}{|BB|}$, which is 1 (so the slices of *Add*(S2) cover 100% of the state space *BB*(S1)). $F(Add,BB)$ is $\frac{|SU(BB)\cap Add|}{|Add|}$, where $SU(BB)\cap Add$ covers the primes at lines $\{6,10,11\}$ (due to data dependency between line 6 and line 11). With this, the Inter-Schema Flow is $\frac{3}{4}$. (Please note that due to the schema inclusion the *Add* schema consists of 4 predicates!) Now, looking at Inter-Schema Coupling, the value is $\frac{1\times1+3/4\times4}{1+4}$, which is 0.8. Similarly, one can calculate the value for the coupling between $BB(S1)$ and $Add2(S3)$, which is 0.833. The new dependency between the invariant at line 6 and line 18 led to the situation that the slice contains one more prime. For similar situations we might infer that the introduction of postcondition primes will (very likely) raise the value of coupling.

Another situation occurs when looking at the operation schemata *AddGift* and *AddGift*2. In relation to the state schema the second variant of the operation contains an additional prime at line 35. However, the point of departure is not the same. In this case the prime is a precondition prime that does not influence any primes outside the schema – at first sight. On closer examination it is clear that the postcondition primes in this schema are control dependent upon this prime, and a slice "reaching" the schema operation will have to include this prime, too. It grows in size, meaning that more "bandwidth" is spent on it. In similar situation we can infer that the value of coupling will also increase as the value for the Inter-Schema Flow increases from $\frac{n}{m}$ to $\frac{n+1}{m+1}$.

The above specification does not show that the value for coupling can also decrease. This is the case when we introduce a postcondition prime that is not related to the primes in the other schema(ta). Then, in case of a state schema, there is no data-dependency between the primes. And in the case of another operation schema there is neither control nor data-dependency. As the size of the schema increases, Inter-Schema Flow decreases from $\frac{n}{m}$ to $\frac{n}{m+1}$.

On the syntactical level there is no difference between *Add* and *AddGift*. Both consist of a precondition prime, three postcondition primes, and include the state. This equivalence can be seen in Fig. 2

as the values for cohesion are the same. But it gets interesting when comparing them to *AddGift*2. The value for Inter-Schema Coupling between *Add* (*S*2) and *AddGift* (*S*5) is 0.750, whereas the value for *Add* (*S*2) and *AddGift*2 (*S*6) is 0.778. So, there is a slightly higher value of coupling between *Add* and *AddGift*2. The reason for this is a semantic difference: the data relationship between the lines 11 and 34. This simple example demonstrates that the idea of the "bandwidth" is quite applicable in this situation.

Though the above example is simple, it is able to demonstrate the effects on the value for coupling in the case of structural changes of schema operations. The second working hypothesis seems to be confirmed, at least from the analytical part of view.

3.3 Limitations

Though the above hypotheses seem to be confirmed, there are limitations. More precisely, the problems are that (a) the specifications might be too dense, (b) only part of the "bandwidth" is regarded, and (c) the dependency reconstruction does not work correctly. What seems to corrupt the measures is in fact not a real problem.

In [2] the effect and efficiency of slicing has been investigated, and it turned out that slicing has disadvantages when the specifications are too dense. In about 60-70% of all cases slicing led to a reduction in the size of the specification, which also means that in some situations there has been no reduction at all. The slices then contained all other primes – indicating that nearly every prime is related to all other primes. However, this effect decreases on average with raising sizes of the specifications (our experience relies on more than 45.000 lines of specification text), and it is only a problem with text-book examples that consist of a view schema operations only.

The next concern is that coupling is not sensitive to changes that lead to an increase in the number of relations between the same primes. Irrespective the number of dependencies between two primes, only the occurrence of the primes is counted by the size-operator. Bandwidth does not increase then. The presented notion of coupling works well on a syntactical level, but not necessarily as expected on a semantic level. The last measure (comparing *AddGift* and *AddGift*2) was sensitive because one prime has (intentionally) been omitted from both schemata: normally, an author of these operations would have added the line $date' = date$ as predicates to the schemata. This would have introduced data-dependencies from both schemata to the *Add* schema, and there would have been no difference in Inter-Schema Coupling anymore. In fact, this can not be seem as a real problem, it is as coupling is defined. Nevertheless, one has to keep in mind that there might be more dependencies as expected.

Finally, slicing only works fine when the specifications are "well-formed". This means that they consist of separable pre- and postconditions primes. When such a separation is not possible, then the outcome is vitiated. Diller mentions in [5, p.165] that in most cases this separation can be done (which means that a syntactical approximation to the semantic analysis is possible), but this does not prevent from cases where pre- and postconditions are interwoven and not separable.

4 Conclusion and Outlook

This contribution introduces the notion of specification slice-profiles that are then used for the calculation of slice-based specification measures. The way of calculating these measures for Z (namely coupling and cohesion) is new and it is based on the use of (reconstructed) control and data dependency information. The objective of this work is to investigate if such a mapping is meaningful. For this, the contribution takes a set of small specifications as a basis, and the sensitivity of the measures is then analyzed by changing internal and external properties of the specifications.

The evaluation shows that the measures reflect the changes in the structure of the specification as expected. Especially the values for cohesion seem to be a good indicator for changes in internal properties. Coupling is, due to the use of unions of slices a bit less sensitive, but it also reacts when there are dramatic structural changes. All in all, the measures proof useful.

The understanding of the *behavior* of the measures was a first, necessary step. The next steps will be to include different "real-life" specifications and to perform an empirical study that demonstrates that the measures are not just proxies for other, eventually size- based, measures. In case of confirming their unique features again, this could be a step towards taking specifications as quality indicators quite at the beginning of the SW-development cycle.

References

[1] Andreas Bollin. *Specification Comprehension – Reducing the Complexity of Specifications*. PhD thesis, Universität Klagenfurt, April 2004.

[2] Andreas Bollin. The Efficiency of Specification Fragments. In *Proceedings of the 11th IEEE Working Conference on Reverse Engineering*, 2004.

[3] Andreas Bollin. Concept Location in Formal Specifications. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(2):77–104, March/April 2008.

[4] Juei Chang and Debra J. Richardson. Static and Dynamic Specification Slicing. Technical report, Department of Information and Computer Science, University of California, 1994.

[5] Antoni Diller. *Z – An Introduction to Formal Methods*. John Wiley and Sons, 2nd edition, 1999.

[6] Mark Harman, Margaret Okulawon, Bala Sivagurunathan, and Sebastian Danicic. Slice-based measurement of coupling. In *Proceedings of the IEEE/ACM ICSE workshop on Process Modelling and Empirical Studies of Software Evolution. Boston, Massachusetts*, pages 28–32, 1997.

[7] S. Henry and D. Kafura. Software structure metrics based on information flow. *IEEE Transactions on Software Engineering*, 7(5):510–518, 1981.

[8] Timothy M. Meyers and David Binkley. An empirical study of slice-based cohesion and coupling metrics. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 17(1), December 2009.

[9] Tim Miller, Leo Freitas, Petra Malik, and Mark Utting. CZT Support for Z Extensions. In *Proc. 5th International Conference on Integrated Formal Methods (IFM 2005)*, pages 227 – 245. Springer, 2005.

[10] Roland T. Mittermeir and Andreas Bollin. Demand-driven Specification Partitioning. *Lecture Notes in Computer Science*, 2789:241–253, 2003.

[11] Tomohiro Oda and Keijiri Araki. Specification slicing in a formal methods software development. In 17^{th} *Annual International Computer Software and Applications Conference*, IEEE Computer Socienty Press, pages 313–319, November 1993.

[12] Linda M. Ott and Jeffrey J. Thuss. Slice based metrics for estimating cohesion. In *In Proceedings of the IEEE-CS International Metrics Symposium*, pages 71–81. IEEE Computer Society Press, 1993.

[13] W.B. Samson, D.G. Nevill, and P.I. Dugard. Predictive software metrics based on a formal specification. In *Information and Software Technology*, volume 29 of *5*, pages 242–248, June 1987.

[14] J.M Spivey. *The Z Notation: A Reference Manual*. Prentice Hall International, 2^{nd} edition, 1992.

[15] Mark Weiser. *Program slices: formal, psychological, and practical investigations of an automatic program abstraction method*. PhD thesis, University of Michigan, 1979.

[16] Mark Weiser. Program slicing. In *Proceedings of the* 5^{th} *International Conference on Software Engineering*, pages 439–449. IEEE, 1982.

[17] Fangjun Wu and Tong Yi. Slicing Z Specifications. *SIGPLAN Not.*, 39(8):39–48, 2004.