

# SE33010 Assignment One - Alexander D Brown (adb9)

## Comparing Vienna Development Model and Z Notation

Both the Vienna Development Model (VDM) and Z Notation (Z) use a model-base specification technique and share a lot of their mathematical notation. Where they differ is in the way their specifications are written[2].

Both are concentrated on the specification of *abstract machines*; a “model orientated” approach. This approach differs from an “algebraic” (or “property orientated”) approaches which focus on defining *abstract data types*[1].

Algebraic approaches give no explicit model of type, defining abstract data types in terms of the relationships between its interactions. In contrast, both VDM and Z give an explicit model of the state of an abstract machine. A commonly used example is a stack; in an algebraic approach it might be defined in a manner described in equation 1, whilst a model orientated approach would typically be modelled as a sequence.

$$\text{pop}(\text{push}(x, s)) = s \quad (1)$$

Where VDM and Z start to differ is in the definitions; VDM has a lot of structure, using keywords to define different parts of a specification. Z has very little of this structure; the definitions are done during the specification and usually consist of *schemas*. These schema define not only the states of an abstract machine, but also the operations.

Z provides a lot more in-built binary relations, this can lead to simpler predicates in specifications as this tends to map to real life data types. However they can also be modelled in a similar way to that of VDM. These Z relations also offer a rich set of operators defined upon them.

Operations yield some of the biggest differences between VDM and Z; syntactically there are quite a few differences; VDM users hooked variables ( $\overleftarrow{abc}$ ) to define variables in the *before* and unhooked ( $abc$ ) for variables in the *after* state.

Z, in contrast, uses undecorated variables ( $abc$ ) for variables in the *before* state, and primed variables ( $abc'$ ) for the *after* state. Z also uses variables ending in ‘?’ for input and variables ending in ‘!’ for outputs.

VDM explicitly defines the ‘pre-’ and ‘post-’ conditions. It also uses an externals clause to define the access to a variable. Z has no equivalent to this clause, the predicate has to explicitly define all final values of variables, even if that variable is unchanged, leading to statements like:  $a' = a$  at the end of predicates.

Another main difference is in the way recursion is handled. There are technical differences in the approaches taken. In VDM the structure (in the case of the example *binnode*) is the new definition, while the type ( $T$ ) is pre-existing. In Z the type ( $T$ ) is the new definition.

Equation 2 [1] shows the Z definition of a binary tree:

$$T ::= nil \mid \text{binnode} \langle T \times \mathbb{N} \times T \rangle \quad (2)$$

Equation 3 [1] shows the VDM definition:

$$\begin{aligned} \text{binnode} &:: l : T \\ &\quad v : \mathbb{N} \\ &\quad r : T \\ T &= [\text{binnode}] \end{aligned} \quad (3)$$

## References

- [1] I. J. Hayes, I. J. Hayes, C. B. Jones, C. B. Jones, J. E. Nicholls, and J. E. Nicholls. Understanding the differences between VDM and Z. Technical report, 1993.
- [2] Ian Hayes. VDM and Z: A Comparative Case Study. 1994.