# SE33010 Assignment Two - Alexander D Brown (adb9)

## Moving from the Spiral Model to Formal Methods

The Spiral model of software development is a lifecycle which is intended for large, expensive and complicated projects, explicitly including risk management as part of the development process.

Formal Methods of software development have many different flavours but focus upon proving that a piece of software satisfies a (formal) specification, typically this specification has a mathematical form. Formal Methods are very useful for safety- or mission- critical systems; where traditional methods would have to rely on large amounts of testing.

It should be pointed out that formal methods do not prove that a program is correct; the methods only focus on satisfying the specification. If the specification is sufficiently detailed to ensure the software is correct then it should follow that the formal methods will ensure the software is correct as well.

Both methods start similarly; gathering requirements for the system. However the Spiral model focuses on both requirements and risk analysis whereas Formal Methods these requirements are used to build an abstract specification.

## The Vienna Development Model

This report will focus on the Vienna Development Model (VDM) as the model for a Formal Development Method, however that are many others with different syntax or structures.

With VDM the two main focuses are on *Data Reification*: the development of abstract data types into concrete ones and *Operation Decomposition*: the development of algorithms from the implicit specification of functions and operations.

The first change that would need to be made is to build this abstract specification based on the requirements, defining the data types and operations which are needed to complete the system. Included in these data types are the data type *invariants*; conditions which can relied upon to be true.

Along with these data types some theory is also developed, for example if a set within the data type must have at least one element in it a theory or *lemma* is defined.

With these data types defined the set of operations which need to be performed on them are also defined. These operations can have both pre- and post- conditions associated with them.

## Issues with Moving to a Formal Development Method

Formal development methods typically require markedly different skill set from traditional software development, a heavy knowledge of mathematics; particularly set theory and logic; is required to be able to write the formal specifications on which code is built to fulfil.

Another major problem will be the programming language used to implement the formal specification: this language should need to be formally proved itself otherwise the software is being built on an unsound base.

One major issue with formal methods is the significant time investment needed to build the formal specification, when done by hand it is a very time consuming process and is very prone to human error. If checked by a tool or 'verifier' then there is the inherent problem of proving that the verifier is performing its job correctly.

It should be reiterated that formal methods should only be realistically considered for software which is either mission- or safety- critical.

## Tool Support

A number of tools exist to support the conversion of VDM and VDM++ to different languages, including Java and C++. VDMTools is one of the leading commercial tools for this area and involves defining the specification in a text format. VDMTools includes parsers to check both syntax and (data) types.

VDMTools also includes ways of performing tests on the specification to ensure that the invariants and operations are actually defined correctly, these act somewhat similarly to how unit testing would be performed.