3 Balancing Future Proofing and Usability

To support coupling between VDM++ and UML inside the VS Code platform, a number of different approaches were weighted against each other and the following information was taken into consideration.

The connection between Modelio and Overture's UML transformation has drifted over time as Modelio has changed its implementation of the XMI standard to the point that importing a VDM model into Modelio only presents the classes of the model with no relationship between the classes, such as inheritance or dependency. Trying to show these relations however leads to a tangled web of connection which does not give an overview of the models architecture. And that is if one is even successful in importing said model, as conflicting EMF formats leads to incompatibility between versions.

well only partly true...

This challenge is the same for other UML tools, as each tool as has a different way of implementing the XMI standard. The standard therefore fails in enabling interoperability between the different tools.

One could choose a specific tool and develop VDM VSCode's UML import/export functions to suit the tools way of implementing the XMI standard. This would mirror what was done for Modelio and would require continually updating the coupling if the XMI implementation of the chosen tool changed.

However, there unfortunately does not exist any suitable UML tool extensions for VS Code that can import a UML file which adheres to the XMI standard. To enable a seamless visualisation of CDs inside VS Code, one would need to create a UML visualisation tool form scratch that supports this functionality, which is not in the scope of this project.

A solution to this is generating UML from a text based language and create a translator from VDM to that language. It then would not depend on that UML tool to have an import function as one could simply output some text or a file that describes the VDM model in the tools language.

One such tool is PlantUML [18] which fulfills the previously above stated criteria as it is a text based diagram tool that exists as a VS Code extension [7]. As an extension it appears as a tab inside VS Code, it has a continually updating live view, and can be worked on by multiple team members as it integrates with versioning tools like Github. It is even cited in books pertaining to critical [3] and cyber-physical [2] systems. These features make the PlantUML extension one of the more relevant options for fulfilling the goals of this project.

This project could choose to ignore the XMI standard entirely and focus on a direct connection between Plant UML and VDM++. Achieving this would result in higher usability that what was achieved with the Modelio connection as users would only need to open a new tab in VS Code as opposed to opening a new program to import or export VDM++ models.

This would however greatly decrease the maintainability of the UML to VDM++ connection and it would not be particularly future-proof. Although the creator