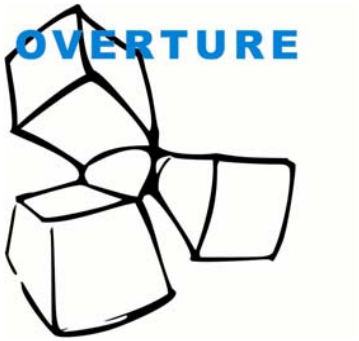
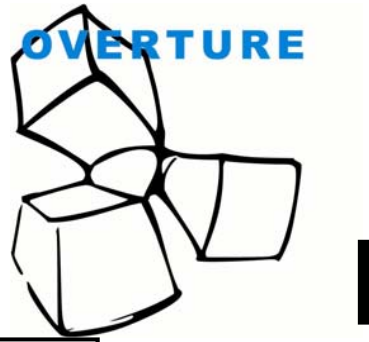



# Brainstorm

Potential subjects (subjects marked red were discussed):

- H2 to fund the Overture initiative?
- H2 improve the architecture?
- H2 integrate parallel developments?
- H2 interest others (students, etc.)?/H2 keep this thing alive?
- H2 establish/maintain the relationship with VDMTools?
- H2 Make VDM++/OML *formal*?
- H2 Get consensus on an overall Overture tool architecture and exchance mechanism?



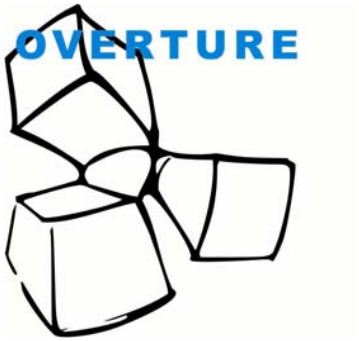

# Wishlist for toolset plug-ins (random order)

- Static semantics checker [Alexander Petrenko, Marcel Verhoef (40% chance on student)]
- Dynamic semantics (interpreter, debugger) [University of Newcastle, Minho University]
- Code generator (Java, C++, C#) [Marcel Verhoef]
- Pretty printing (with test coverage)
- Connection to UML tools [Araki San]
- Support for test cases [Bernhard Aichernig]
- Proof obligation generator [University of Newcastle, Bernhard Aichernig]
- Proof support [University of Newcastle, Araki San]. On the proof subject there is existing work (e.g. PROSPER). H2 to get hold of this?
- Real-time support [Marcel verhoef]
- Extended static semantics checker (e.g. for concurrency aspects) [Marcel Verhoef]
- API support [Alexander Petrenko]
- Refinement + refactoring [Minho University]
- Reverse engineering [Minho University, Delft University of Technology?]

Action: find out which group already works on which subject and approach them actively

Green text indicates *interest* in the subject only

Static semantics checker and dynamic semantics have priority because some other components cannot be made without them




# Funding possibilities

- Companies that benefit from Overture (e.g. also companies interested in additional Eclipse plug-ins)
- Research projects (but there has to be something “new” in the proposal then
- FME (can only count on small amounts)
- Framework 7
- Organisations that stimulate open source developments (perhaps talk to Eric Verhulst)


# Other remarks/actions

- Look into Scala
- Put more information on the web site (documents, minutes of meetings, etc.)
- Keep Overture components small (easier to maintain)
- Start using the tools ourselves
- Announce projects for component development on the web site to attract students (such projects must follow the Overture guidelines)
- Make a list of relevant research questions
- Make a “contributions guidelines document”
- Make an “Overture whitepaper”
- Describe the Overture foundations
- Split the language into a “core” and a version with extensions (easier to work on for students)
- Establish a structure for the project (e.g. With a chief architect role and a language board)
- Use a different name (not VDM++ or OML), e.g. V# or VDM#?)
- Investigate possibilities to participate in a students exchange programme (students like to travel)
- Define extension points at an early stage