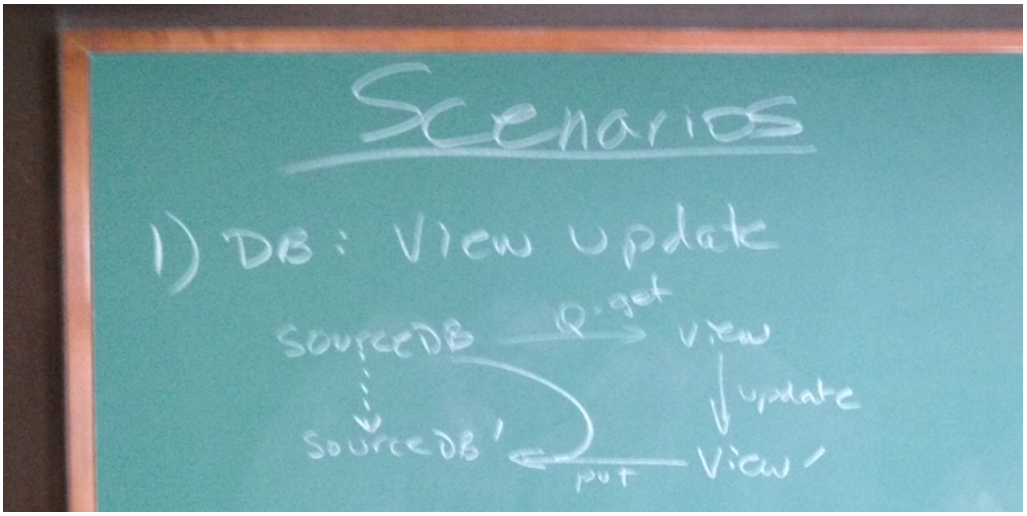
Scenarios for Bidirectional Transformation

The working group met on 3 December to try to identify distinctive *scenarios* for BX from different disciplines, and to attempt to identify key characteristics of a BX scenario. The following scenarios (some of which were contentious!) were identified and discussed. The main points of contention seemed to orient around whether the problem scenarios were inherently bidirectional, or whether the solutions were bidirectional.

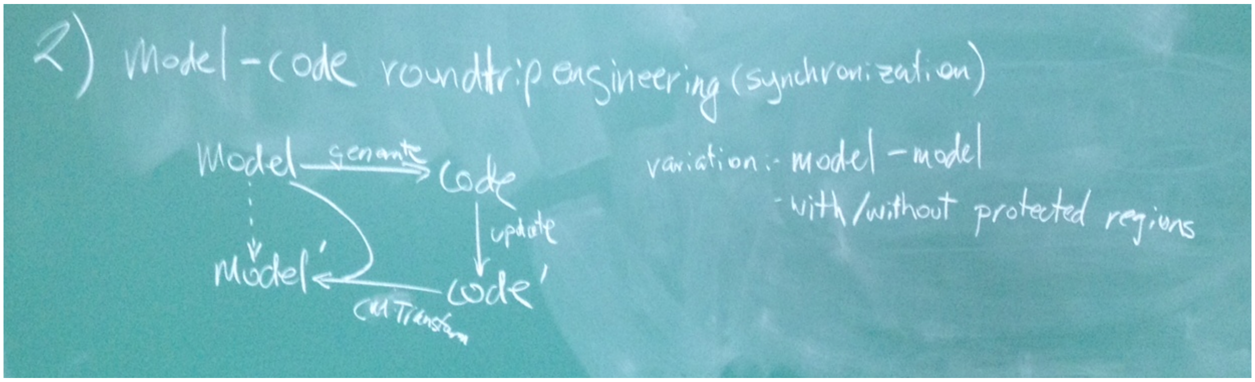
1) Database view update



2) Synchronisation of artefacts

Two variants were identified:

1. Model-code synchronisation, wherein changes to either a model or generated code would be reflected in the opposite artefact.
2. Model-model synchronisation, wherein changes to either “source” or “target” model would be reflected in the opposite model.

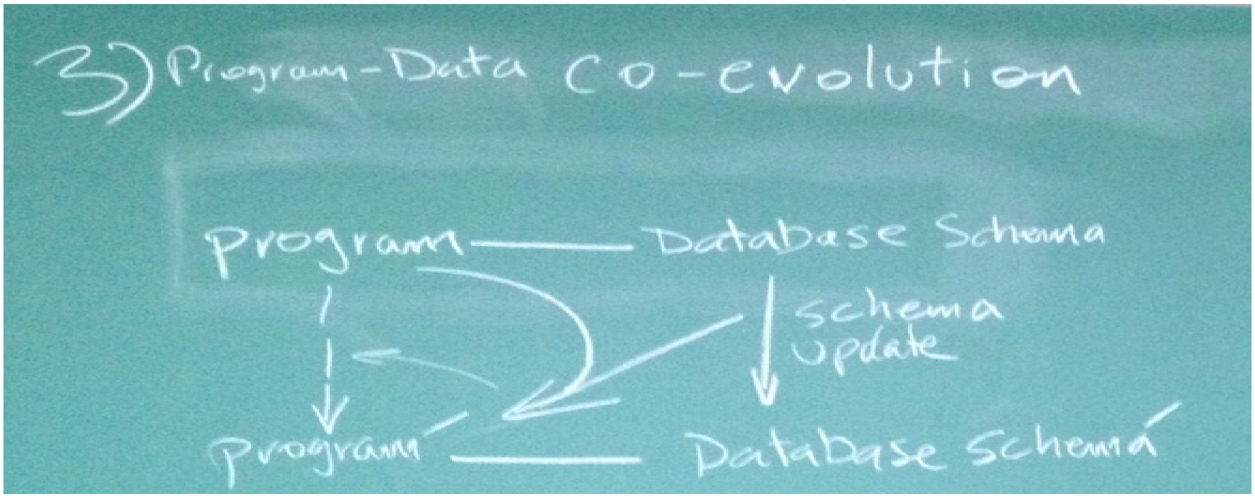

2) Model-code roundtrip engineering (synchronization)
Model
generate
Code
update
Model'
Code'
CM Transform
variation: - model-model
- with/without protected regions

3) Artefact co-evolution

Two variants were identified:

1. Program-data co-evolution, where a program makes use of data (via a database schema), and either program or schema are updated.
2. Model-metamodel co-evolution

There was later discussion that suggested this is a BX problem only when you are concurrently using different versions of the database schemas/metamodels.

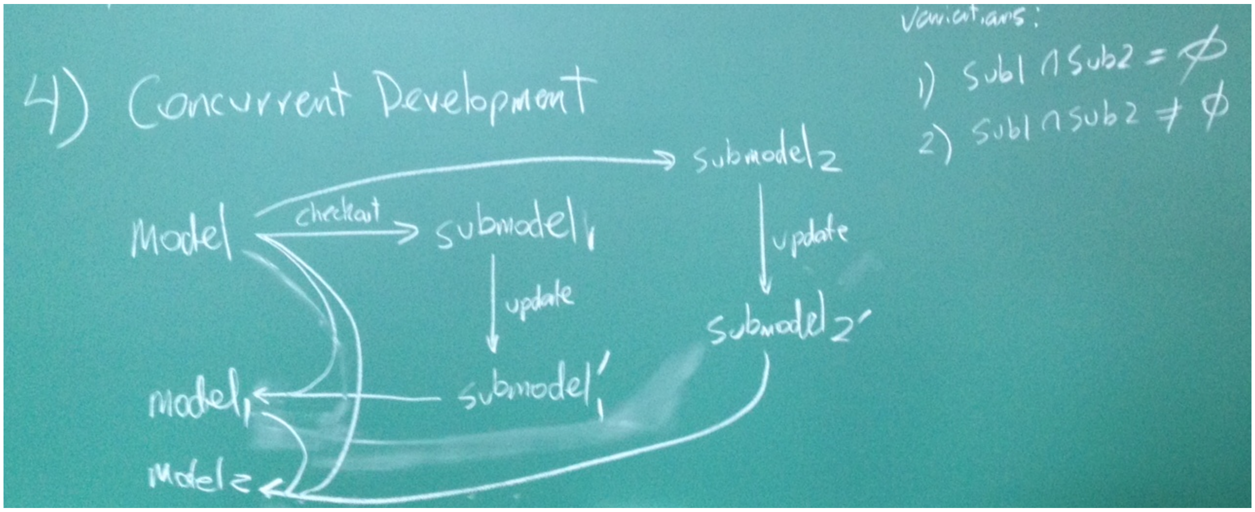


4) Concurrent development

Where a very large model is been operated on by concurrent distributed teams, each of which “checks out” a submodel, makes updates, and commits the changes (which may be in conflict) to the large model.

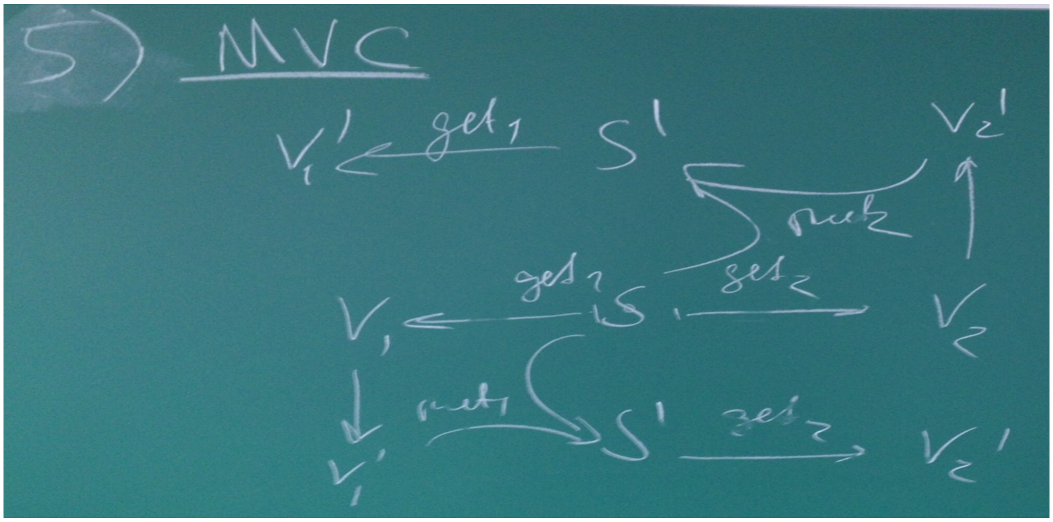
Two variants were identified:

1. Where the checked-out submodels do not overlap
2. Where the checked-out submodels do overlap.



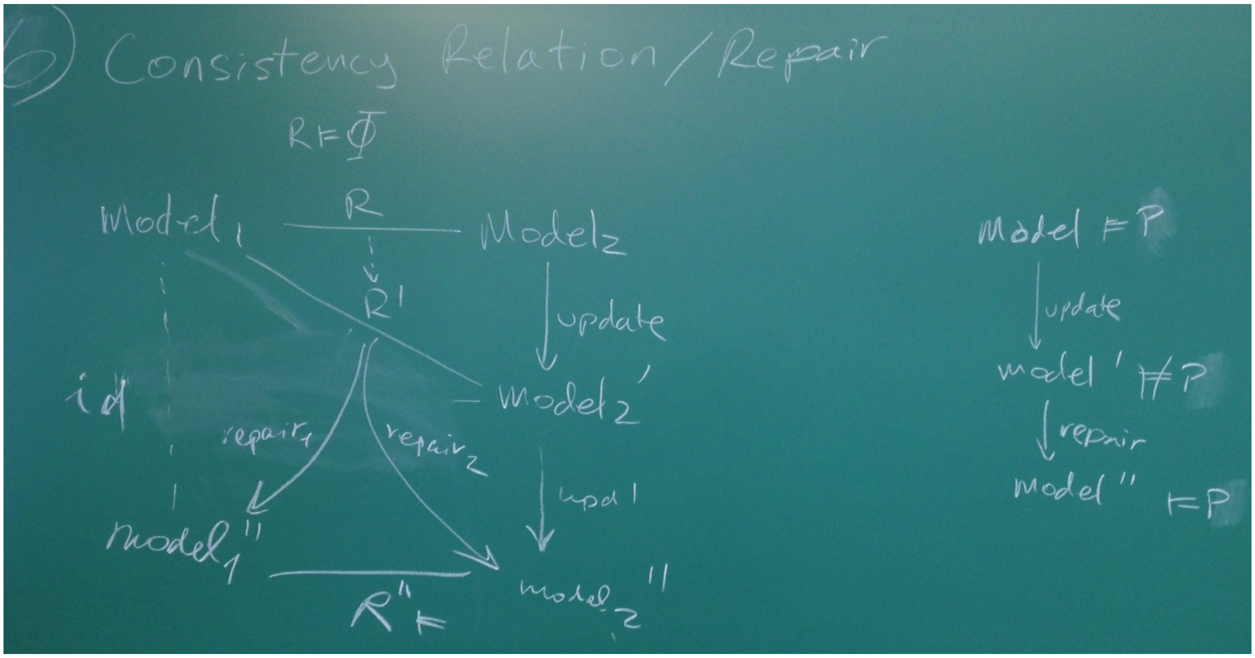
5) MVC-style development

Where concurrent updates to “views” of models can be made, with a controller-like mechanism used to resolve inconsistencies in models (e.g., two-phase commits). A concrete example is where you have Eclipse views (panes) where you make changes to the underlying “model” in each view.



6) Consistency relation/repair

When a consistency relationship between two or more models is violated and the models are repaired (perhaps concurrently).


6) Consistency Relation/Repair
R ⊨ Φ
Model1
R
Model2
R'
update
model2'
id
repair1
repair2
upd
model1''
R'' ⊨
model2''
model ⊨ P
update
model' ⊭ P
repair
model'' ⊨ P