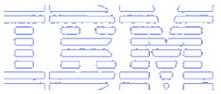
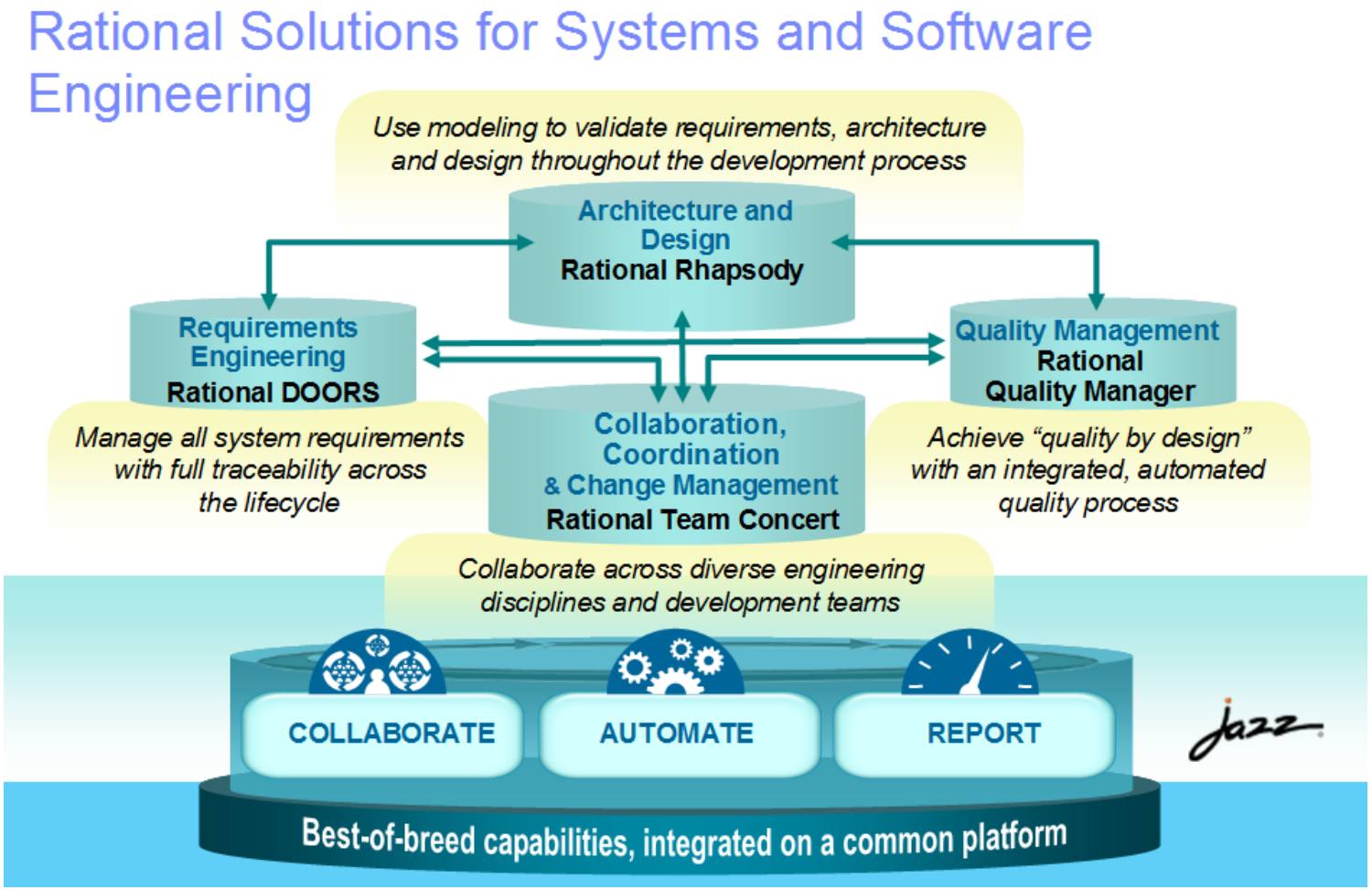
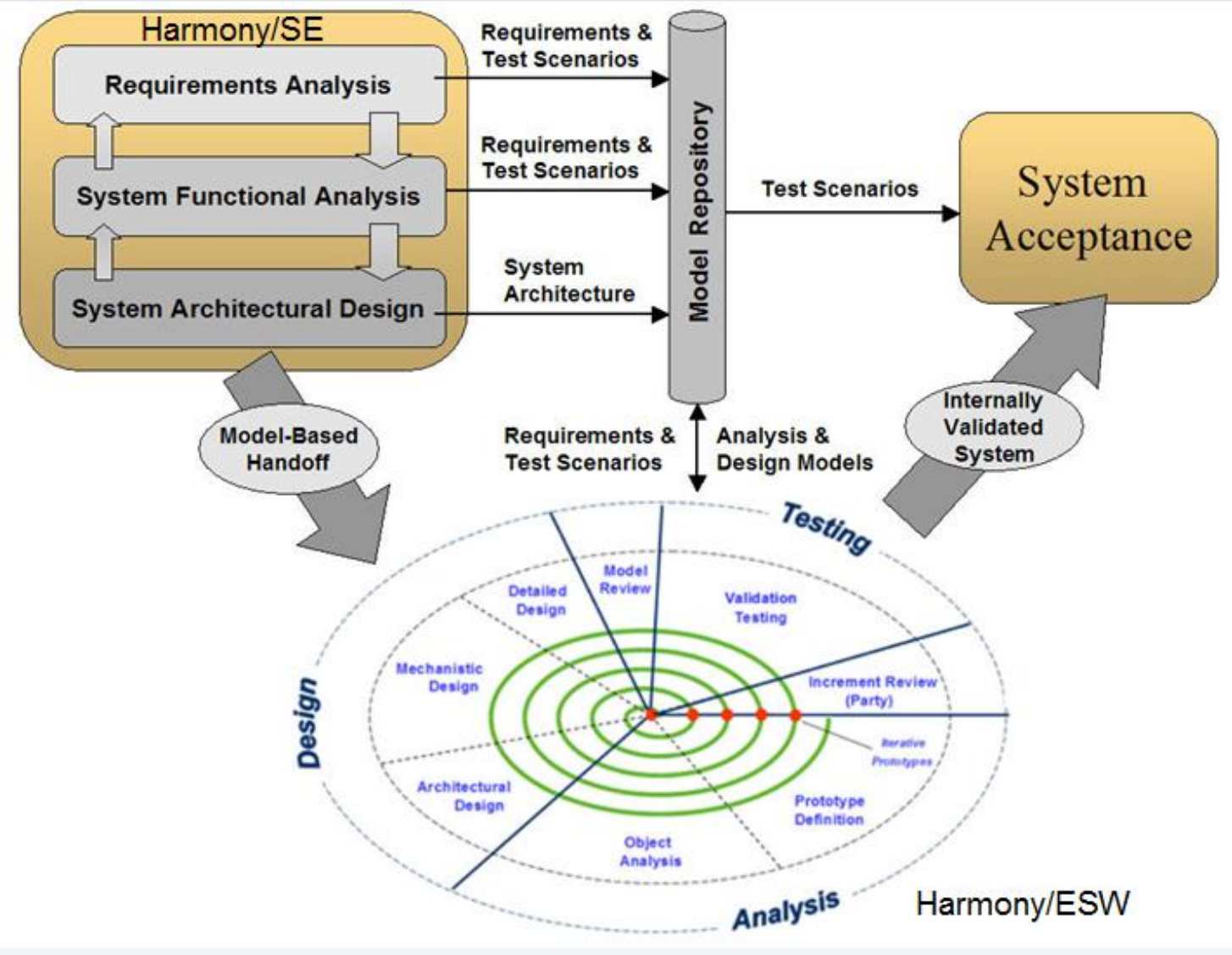


MODEL DRIVEN DEVELOPMENT IN RHAPSODY

Rhapsody & CLM

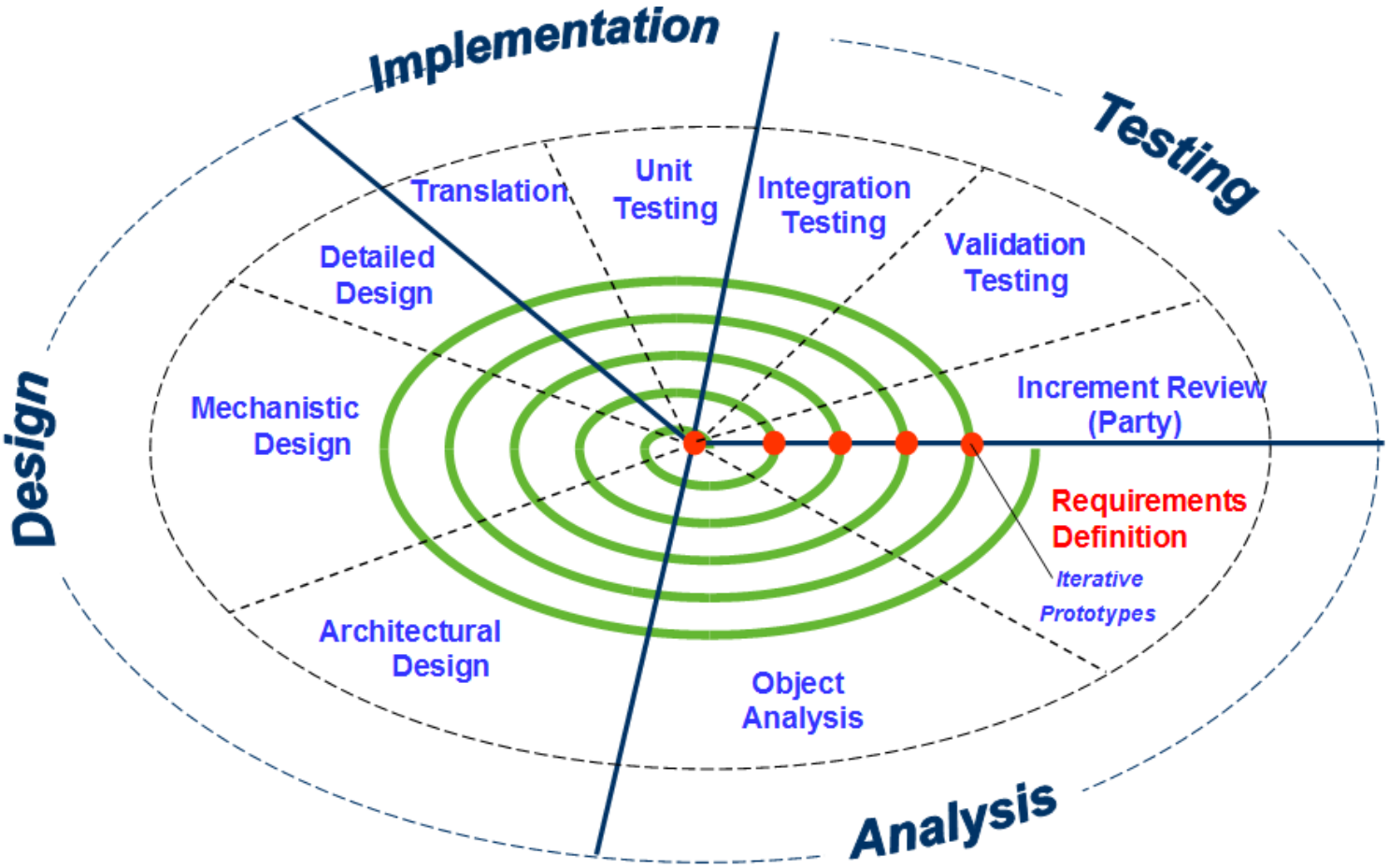


Harmony Process

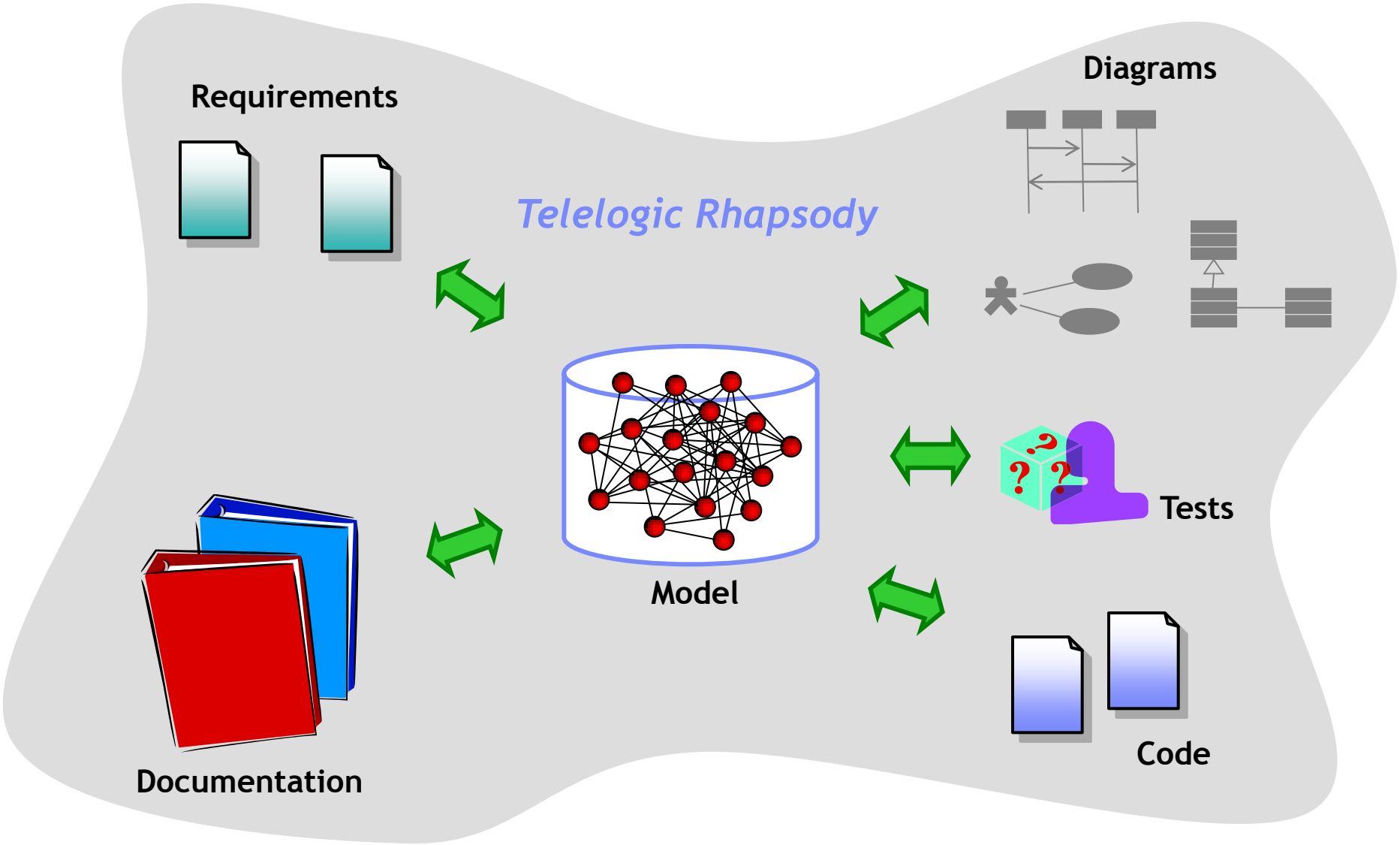


Harmony ESW for MDD

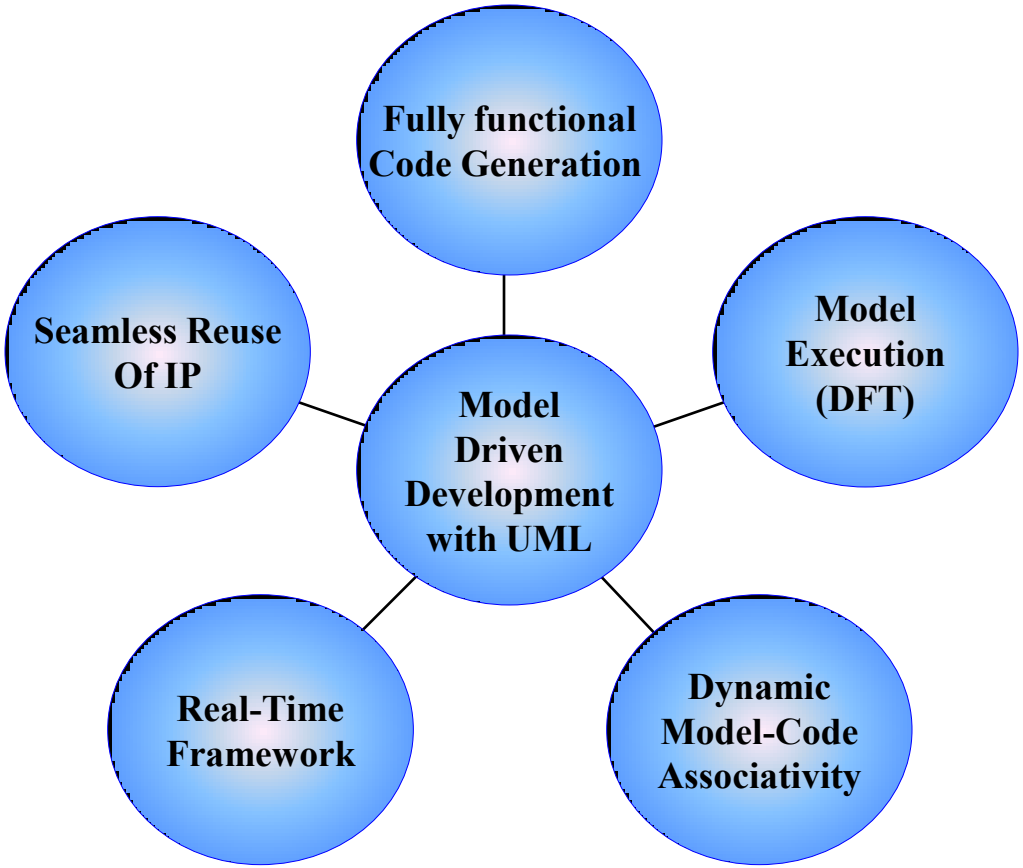
Harmony MDD is an agile based iterative, incremental development process which has a running / deployable prototype at the end of each iteration.



Model-Driven Development

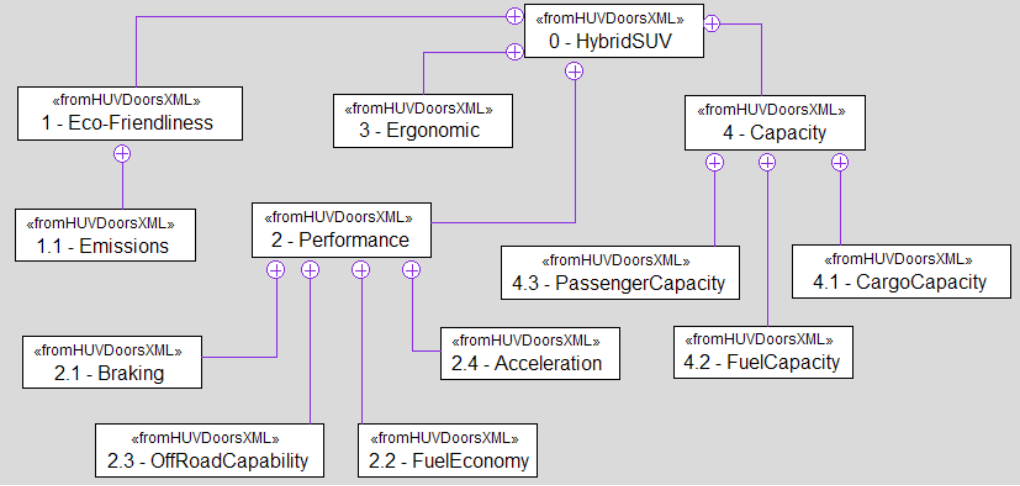


Rhapsody's key enabler



SysML

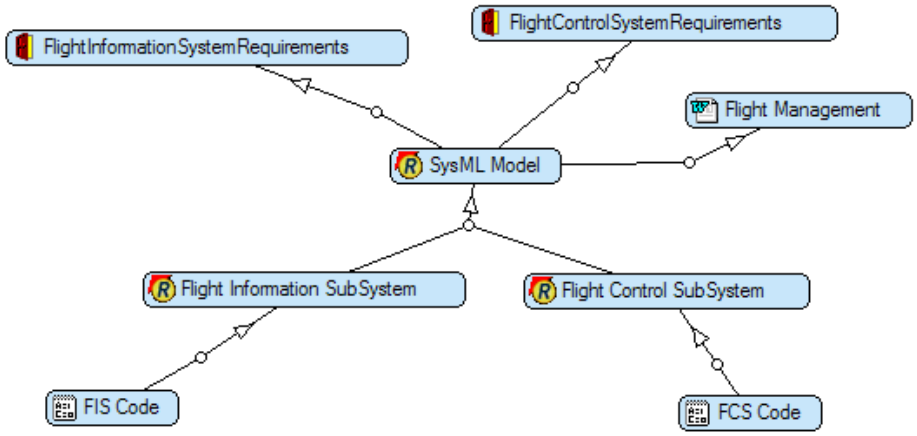
- SysML is a domain customization of UML 2 for systems engineers
  - Supports the standard proposal in its latest form (V1.0)
- Support for SysML views
  - Requirements: Requirements diagram; Use case diagram
  - Structure: Block Definition diagram; Internal Block diagram
  - Behavior: Statechart; Activity diagram; Sequence diagram
  - Constraints: Parametric diagram
- *Uniquely Integrated Requirements and Design modeling environment*
- More than just modeling…
  - Simulation of SysML models
  - System testing for SysML

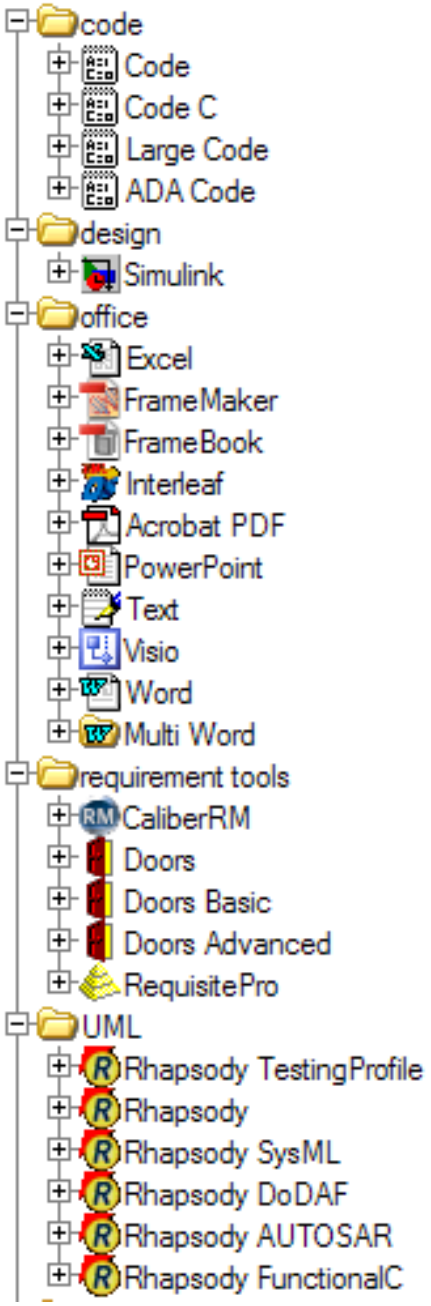


MENU

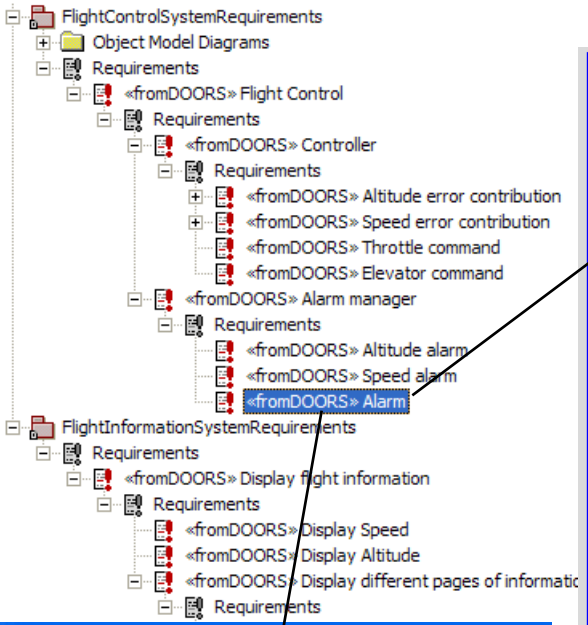
Requirements Modelling

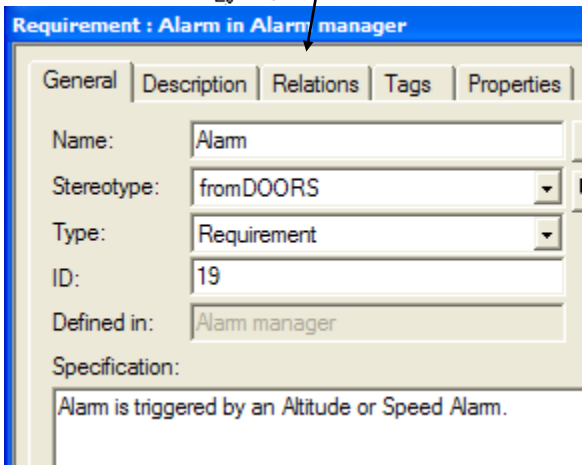
- Requirements Capture
- Requirements Traceability
  - Create traceability links from model to requirements
  - Automatic traceability documentation
- Requirements Analysis
  - Requirement Coverage Analysis
  - Change Impact analysis
  - Automatic report generation

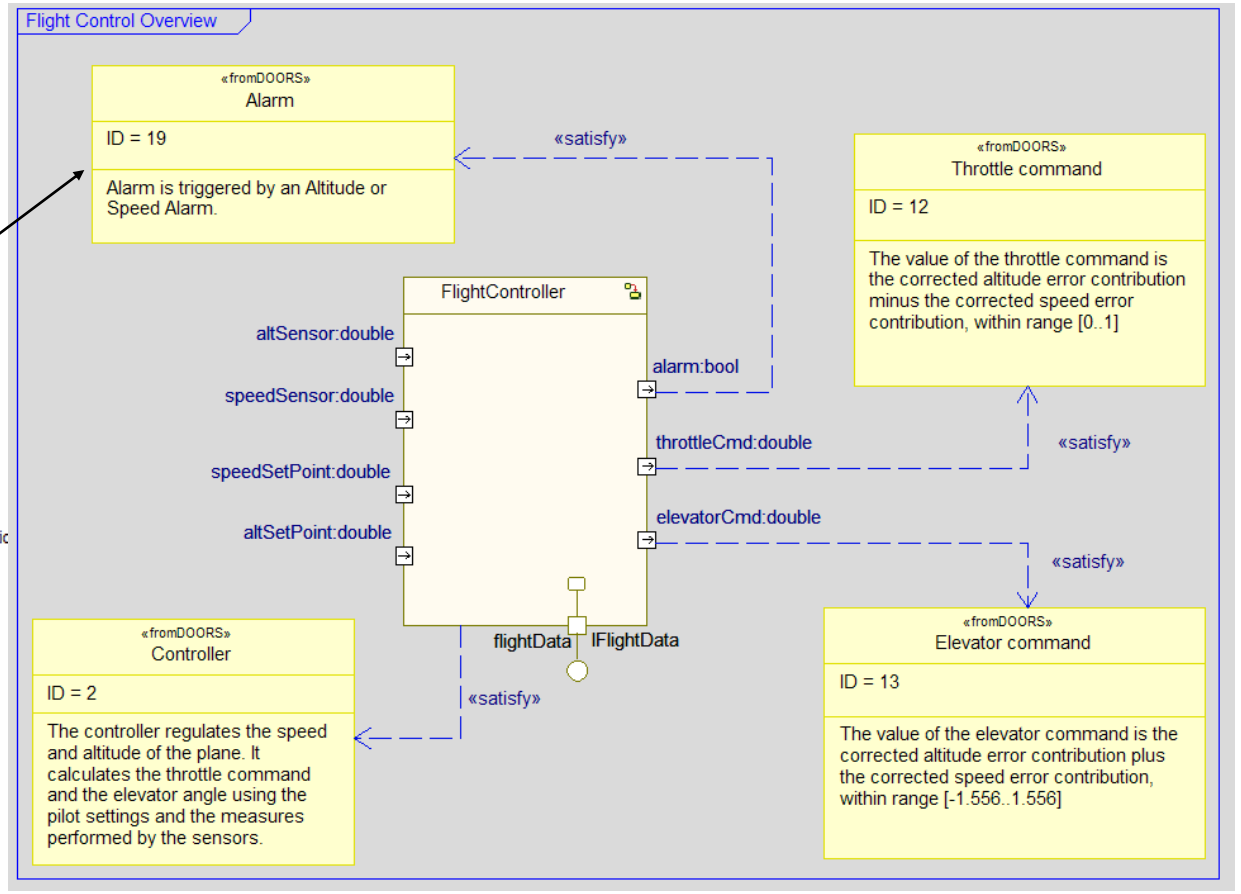




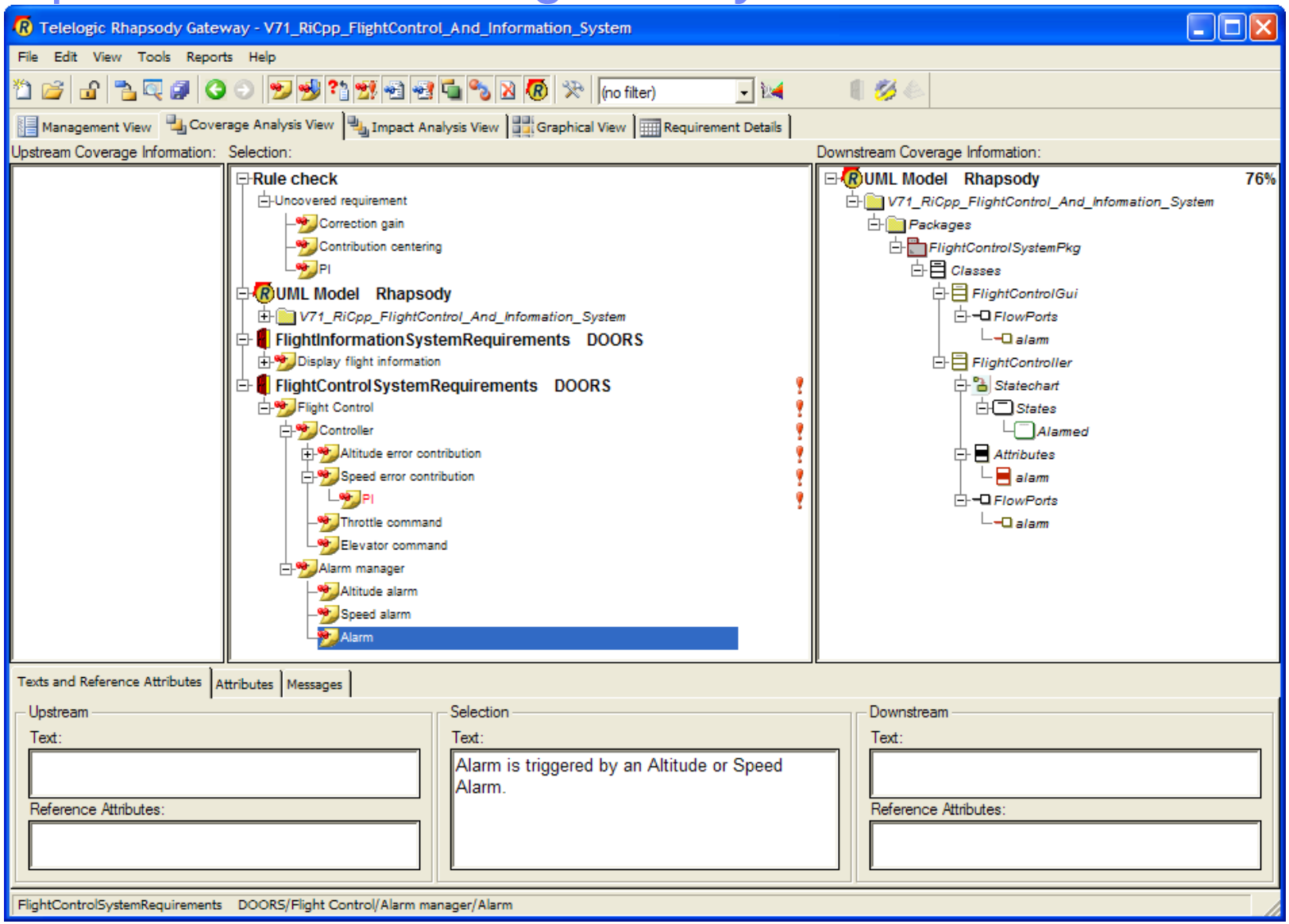
Requirements Capture and Trace







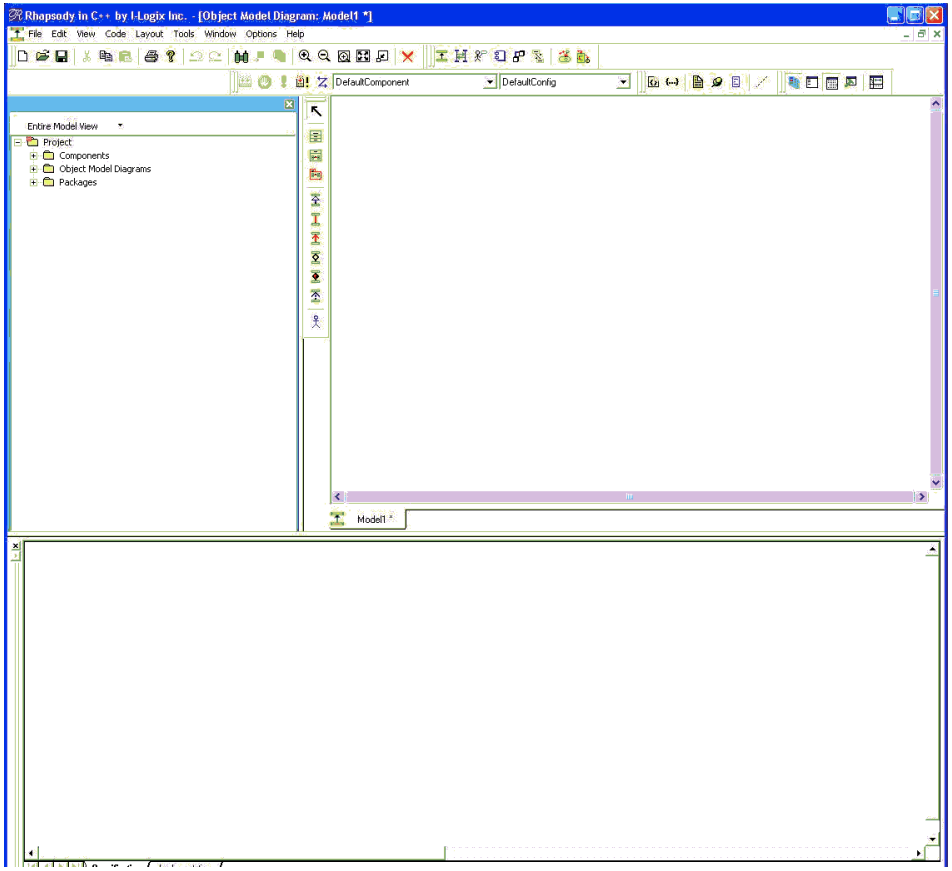
Requirements Coverage Analysis



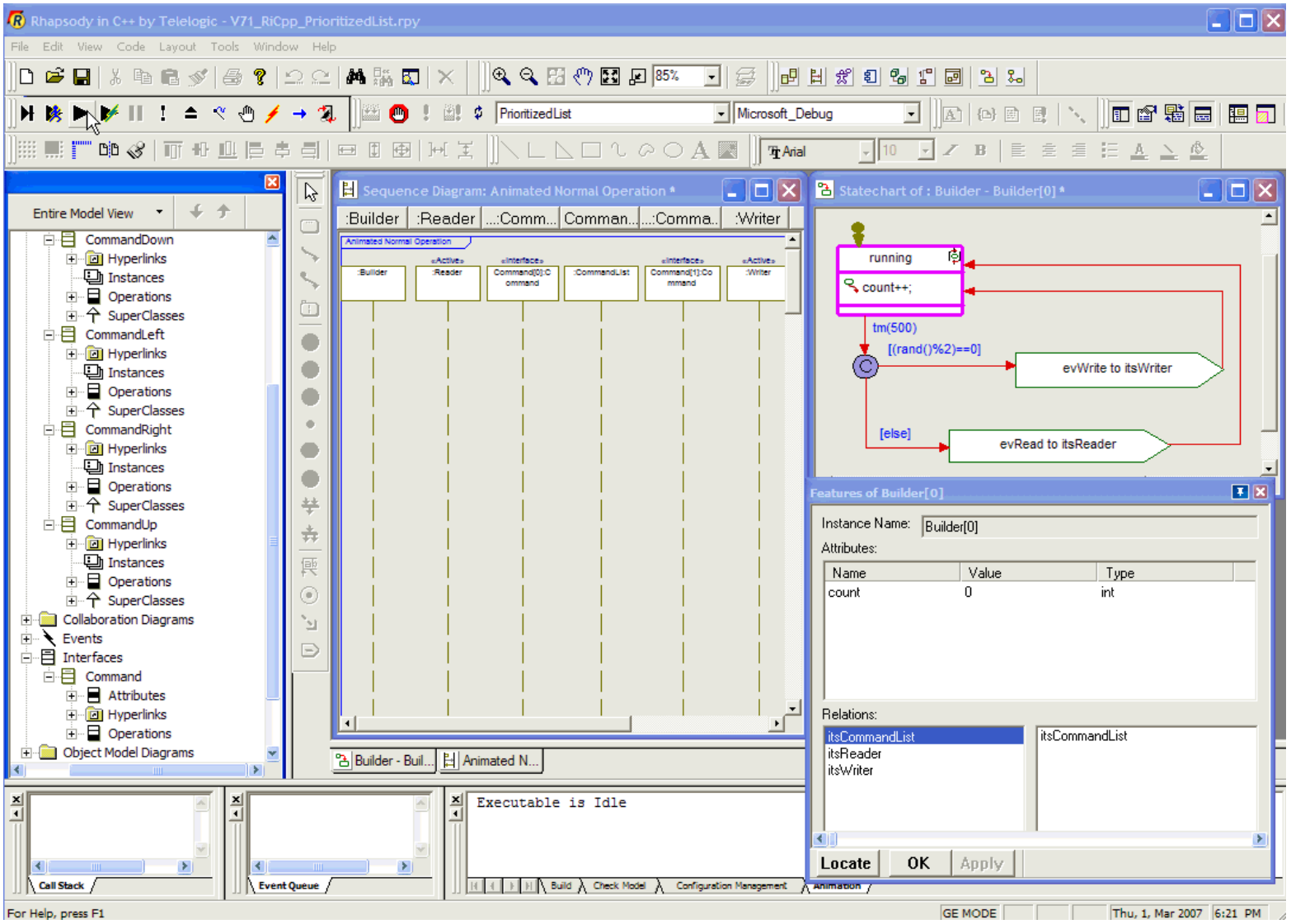
MENU

Dynamic Model-Code Associativity (DMCA)

- Code is another view of the model
- Dynamic bi-directional synchronization of model & code
- Model & code always in sync
  - Direct reflection of model domain into the code
  - Flexibility without separation of model & code
- Increases productivity

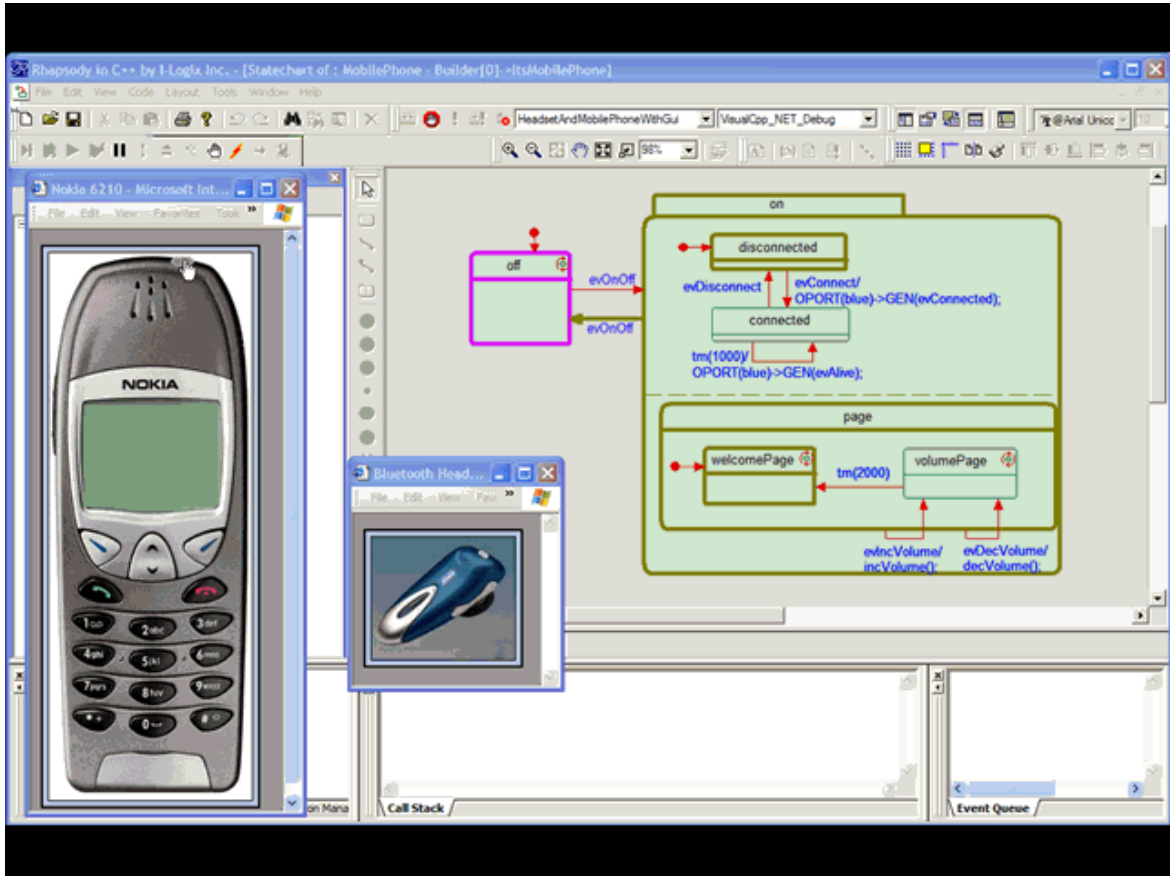


DFT : Executable Models on Host & Target



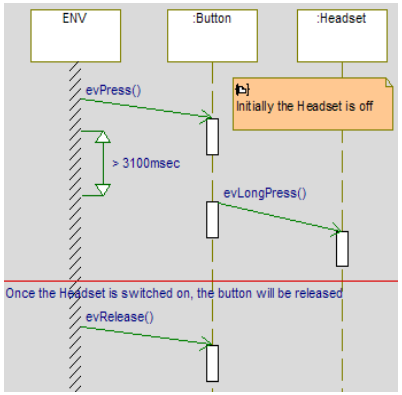
DFT: Collaborative Debugging

- Web based collaborative debugging
- Rapid prototyping

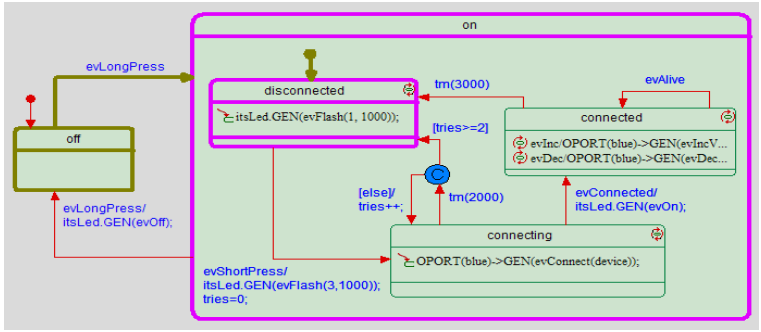


DFT: Test Conductor™

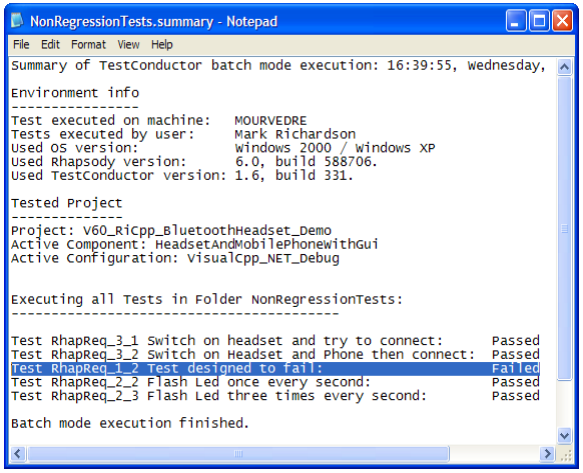
Sequence Diagrams



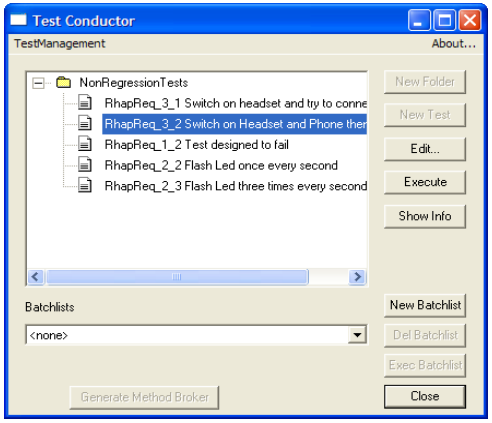
Stimulate & Monitor the Model



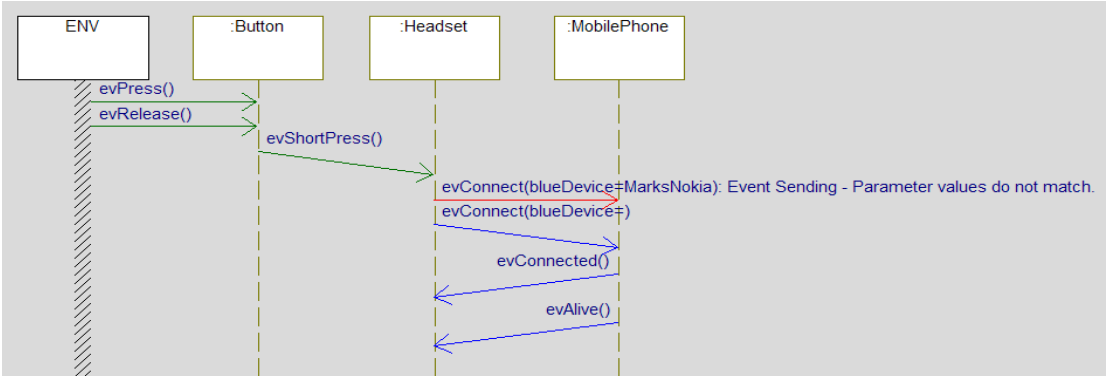
Test Results



Test Configuration

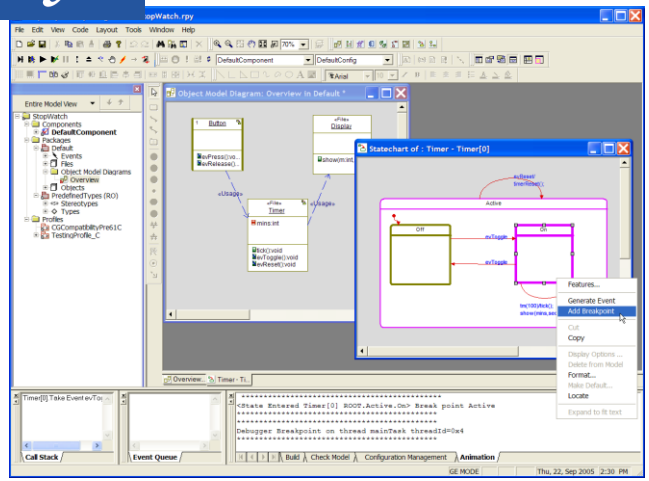


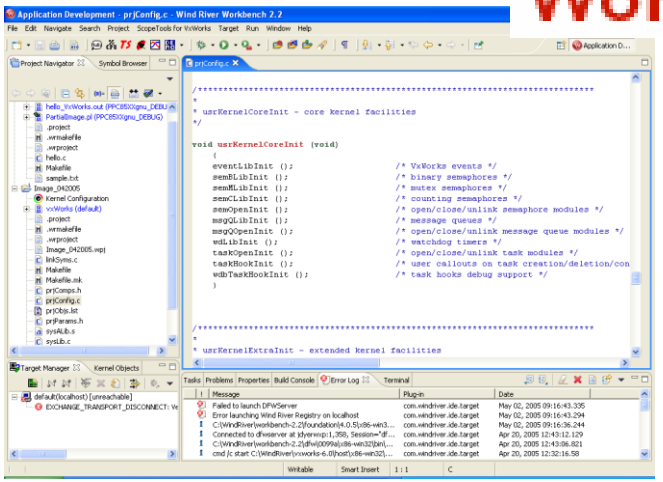
Finding & Correcting Errors



Rhapsody®

Workbench





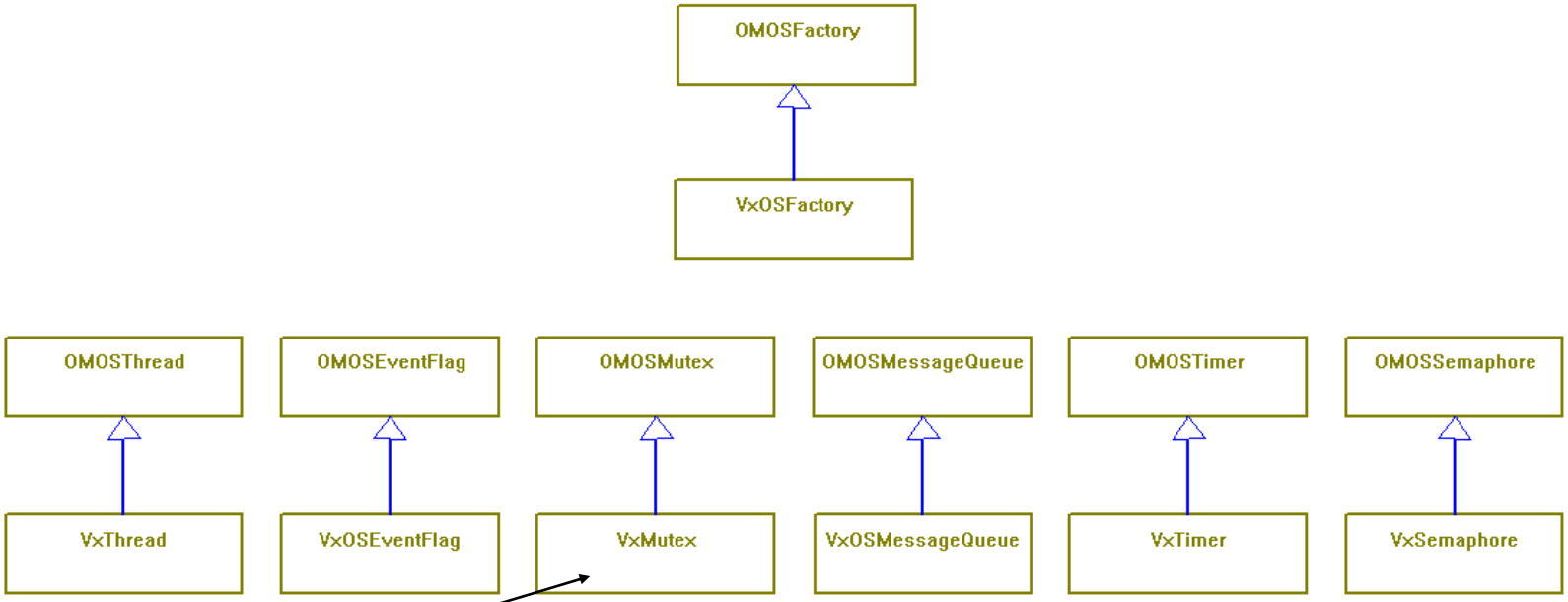
MDD
Code Generation C, C++ Ada
Combined source and Design-level debugging

Automatic Download
Synchronized Breakpoints
Unit Testing

Targets

MENU

VxWorks Adapter

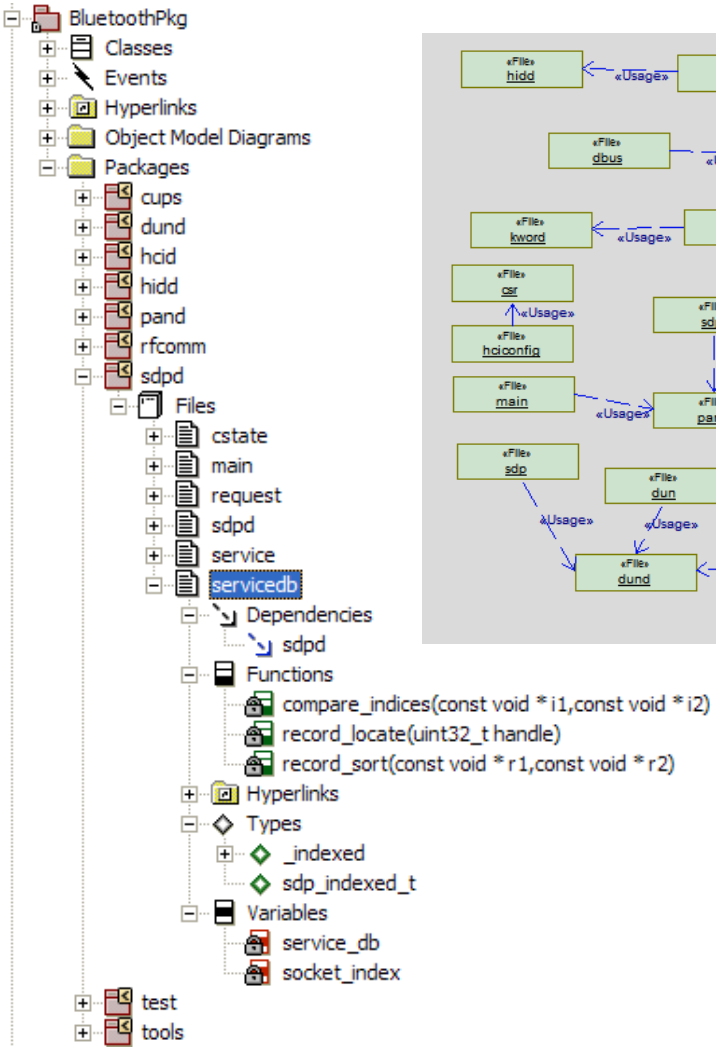


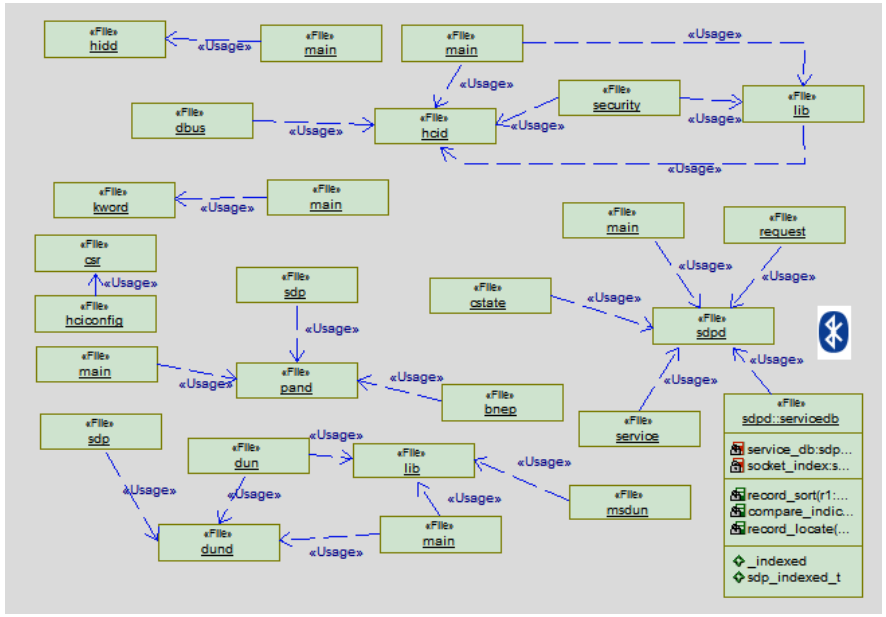

```
class VxMutex : public OMOSMutex {
private:
    SEM_ID  hMutex;
public:
    void lock()   { semTake(hMutex, WAIT_FOREVER); }
    void unlock()   { semGive(hMutex); }
    VxMutex() {
        hMutex = semMCreate(SEM_Q_FIFO);
    }

    ~VxMutex() {semDelete(hMutex);}

    void* getHandle() { return (void *)hMutex; }
    virtual void* getOsHandle() const { return (void*) hMutex; }
};
```

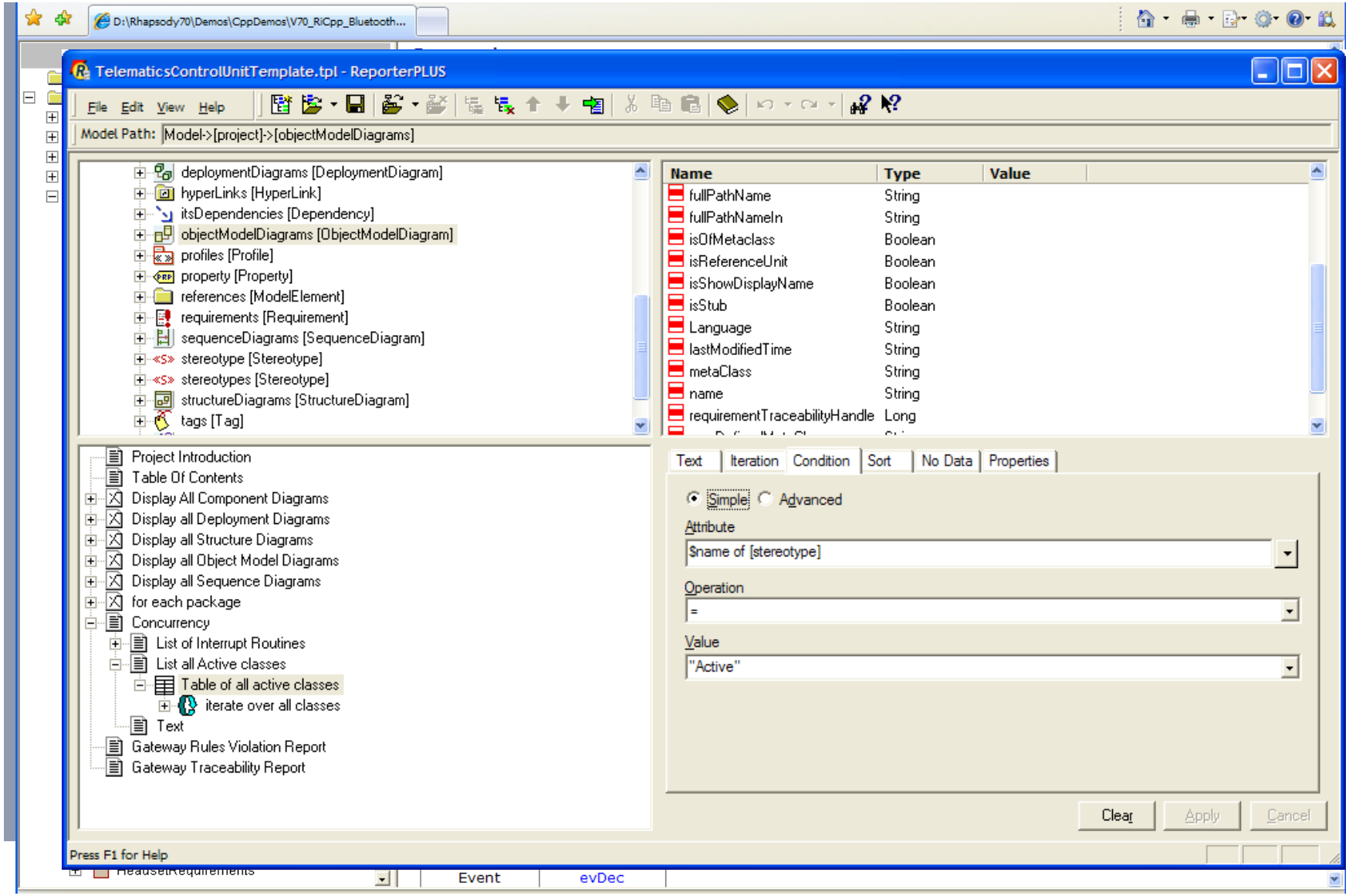
Reuse of IP : Import Legacy “C” code





MENU

Documentation: Rhapsody ReporterPLUS™



MENU