The Omega IST project

Model based development and use of formal methods in the context of real-time software

http://www-omega.imag.fr/

Partners

Academic (tool and technology providers)

- Verimag, France – coordinator
- Christian-Albrechts University Kiel, Germany
- CWI (Centrum voor Wiskunde en Informatica), Netherlands
- University of Nijmegen, Netherlands
- OFFIS, Germany
- Weizmann Institute, Israel

Users

- EADS Launch Vehicles, France
- France Telecom R&D, France
- Israeli Aircraft Industries, Israel
- NLR (Nationaal Lucht- en Ruimtevaartlaboratorium), Netherlands

Supporters (CASE tool providers)

I-Logix --- Rational Software, IBM --- Telelogic

Model based development in the context of real-time and embedded systems

General ideas and derived requirements:

- A model integrating different aspects of the system (and its environment)
  - ➔ Possibility to represent different aspects of heterogeneous systems
- Maintenance of a consistent model throughout the development
  - ➔ A semantic framework consistently integrating all aspects
- Early detection of design errors by realistic simulation and testing at early stages of design
  - ➔ Existence of an operational semantics even for abstract high level models
  - ➔ Take into account non functional aspects early
  - ➔ Early formal validation

Analysis of real-time systems today

Current practice in a model-based approach (oversimplified):

Step 1: Build a functional model, analyse and refine it until stable

Step 2: Independently (or almost) of the functional model, build a task model and do timing analysis based on simulation or analysis tool (mainly RMA)

Problems:

- risk of inconsistency between functional and task model
- if time analysis reveals problems, step 1 has to be started all over again
- modification in step1 of the models increases the risk of introducing inconsistency

Development and verification in a model based approach

Step 1: Build an initial model of the system and its environment including both functionality and relevant timing information

Step 2: **Extract** several models and analyze them using formal techniques:

- A model focussing on functional correctness: use untimed verification to detect deadlocks, unreachable states, …
- A model focussing on timing : use timed verification tools to detect timing errors, race conditions, …
- …

Step 3: Modify and refine the initial model, verify refinement formally, and redo step 2

Formalisms for expressing real-time

Verification methods and tools for real-time systems developed by the formal methods community

- **Good semantic level formalisms** for the representation of models including timed aspects (extensions of timed automata, …)
- **Verification and analysis** tools for these formalisms (symbolic analysis, model exploration based analysis, theorem proving)

Problem: **low level representation** of real-time systems,

- convenient for representing some extracted model for timed verification
- not convenient for modeling time at user level

Problems addressed in OMEGA

Modelling real-time and embedded systems in UML

Problem:

UML lacks sufficiently expressive notations

- for the definition of a ***functional model*** of a software system and its environment including heterogeneous components (different execution and communication modes)
- for defining ***time extensions***
- for the expression of ***requirements*** to be verified on the model (functional and time related properties)

. . . and especially the ***meaning*** of notations

Problems addressed in OMEGA

Verification of UML models

- Problems related to UML
  - Lack of a consistent semantic model for different UML notations
- Problems related to existing verification methods and tools
  - Some UML concepts cannot be expressed in the formalisms of existing validation tools (dynamic systems, inheritance, …)
  - Existing validation methods can not deal with these concepts efficiently (scalability)

    → Compositional and abstraction based methods must be further developed

Problems addressed in OMEGA

Make results available to users of UML CASE tools

- Problems related to deficiencies of UML and Case tools
  - XMI is the standard model exchange format for UML, but
    - It does not cope for all parts (action language, OCL)
    - XMI export is not provided by all tools, and some concepts are represented differently by different tools
    - CASE tools do not implement all notations or impose restrictions on their use
- Problems related to semantic differences with existing case tools
  - Some case tools have nice facilities for interactive model exploration, but they are based on a particular tool semantics

Omega project: a proof of concept

1. A subset of **UML notations** for the representation of models (class diagrams, state charts, architecture and component diagrams, real-time profile) and requirements (LSC, OCL)
   - Extensions for sufficient expressive power
   - A semantics integrating all notations consistently
2. Adaptation of existing **validation tools** for the validation of UML models by mappings from UML (XMI) into input format of the existing tools by respecting the defined reference semantics
   - Extensions of internal formalisms to cope with the expressive power of UML
   - Improvement of existing validation methods
   - Development of compositional verification methods based on the components concept
3. A **methodology** for the use of the defined notations and tools
4. **Evaluation** of the developed tools and methods by means of case studies provided by industrial users