AUTOMATIC ANALYSIS OF UML MODELS

Goals

Our approach aims at the extension of modern system design methodologies, like CASE and hardware-software co-design, by automatical, model based mathematical analysis and validation.

UML

The most important object-oriented CASE and HW-SW co-design tool providers joined recently to construct and support a new system design language, the Unified Modeling Language (UML), which is currently being standardized at the Object Modeling Group (OMG). UML is a flexible modeling paradigm:

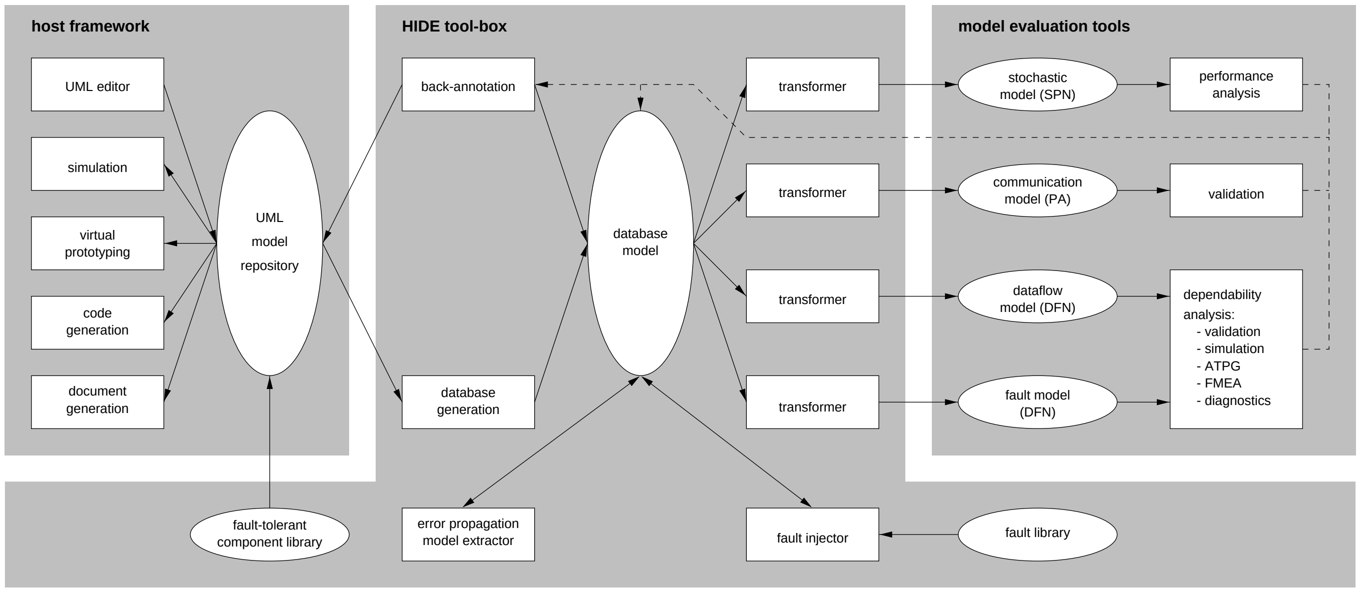
- covers the entire functional design process;
- comprises a variety of formalisms required for the easy description of design solutions;
- close to the engineers' way of thinking;
- supports parallelism and hierarchical model refinement;
- enables the re-use of designs and solutions.

UML is widely used for software development and for the design of embedded systems. Code-generators exist for numerous languages like C++, Java, VHDL, SQL etc.

Proposed solutions

We propose solutions for the following problems of the design process:

- **Formal verification based on mathematical models.**
  Simulations and experiments/measurements on (virtual) prototypes are insufficient for the *proof* of correctness.
- **Model-based dependability analysis of the system.**
  Dependability is a key design factor in modern systems which are increasingly complex and sensitive to disturbances of the environment, but are required to provide continuous service.
- **Analysis of performance and performability.**
  Even advanced design methodologies still lack the proper support of performance analysis, which often results in costly re-design or post-upgrades of the product.
- **Analyses based on the model being used in the design process.**
  In this way the difficulties originating in the necessity of simultaneous use of multiple, different mathematical models for the analysis can be avoided.



The UML-based automatic model analysis environment

Automatic model analysis

Proper quality of the services delivered by the product has to be assured during the design process also by formal methods and mathematics.
By using automatic model transformations the need for both a specific expertise in abstract mathematics and the tedious manual re-modeling of the system for analysis can be avoided:

- the practitioner designer is allowed to use UML as front-end for the specification of both the system and the user requirements;
- from the basic UML model *automatic transformations* derive the individual mathematical models for formal and quantitative validation;
- the results gained by the analysis are back-annotated for presentation into the same UML model without any interaction of the user;
- thus the engineer can validate the design easily, since the entire background mathematics are hidden from him.

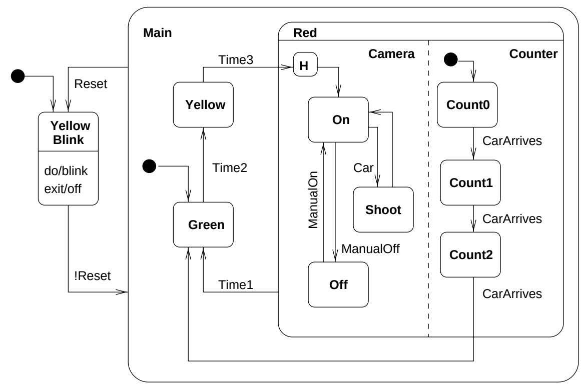
By our approach existing tools and analysis methods can be incorporated into an integrated, UML-based design environment.

Formal verification

Formal verification enables to answer questions about the behavior of the system like

- freedom from deadlocks;
- avoidance of unsafe states;
- guaranteed service of requests without losses;
- existence of dangerous – or desirable – state sequences.

Formal verification is based on the transformation of UML statechart diagrams. In the background efficient tools (like SPIN or VIS, being capable of handling over 10^{20} states) are applied.



Dependability analysis

Model-based dependability analysis can support optimal decisions in the design process by

- comparing design alternatives;
- finding the dependability bottlenecks of the system;
- identifying (by sensitivity analysis) the critical components.

The analysis is based on UML class, object and deployment diagrams.

Performance analysis

Performance analysis can answer the following questions:

- time spent in given states of the system;
- time required to execute a given sequence of actions (e.g. service of a request);
- execution (or occurrence) rate of given events (e.g. the use of a given resource).

The performance analysis is based on the behavioral diagrams of UML. In the background stochastic Petri-nets are solved.

Qualitative fault analysis

The qualitative fault analysis examines the potential error propagation paths among software and/or hardware components of the system, and identifies the critical classes, objects and shared resources. Based on this analysis, the designer can decide on the necessary input/output checks and access restrictions.

The HIDE consortium

The solution of the above tasks was elaborated in the framework of the EC-supported ESPRIT LTR project Nr. 27439 "HIDE - High-level Integrated Design Environment for Dependability". The quality of the analysis methods is guaranteed by the well-known research institutes (University of Erlangen, Germany; Technical University of Budapest, Hungary; Pisa Dependable Computing Center, Italy) and companies (MID GmbH, Germany; Intecs Sistemi, Italy) participating in the project.
Alvicom Ltd. (a partner of TU Budapest) offers a C++ code generator and debugger for the UML design tool Innovator.

Contact person: András Pataricza, Ph.D, associate professor e-mail: pataric@mit.bme.hu
H-1521 Budapest, Műegyetem rkp. 9. phone: +36 1 463 3595 fax: +36 1 463 2667