Final Report

Part 1: FIRST PHASE PROJECT OVERVIEW

Esprit Project 27439 - HIDE

***H**igh-level **I**ntegrated **D**esign **E**nvironment for Dependability*

W. Hohl

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

Consorzio Pisa Ricerche - Pisa Dependable Computing Centre (PDCC)

Technical University of Budapest (TUB)

MID GmbH

INTECS Sistemi S.p.A.

Contents

1. Results of the first phase

1.1. Research results with regard to the objectives

The main objective of HIDE is the creation of an integrated environment for the practitioner designer, allowing him to use UML (Unified Modeling Language) as front-end for the specification and analysis of both the system and the user requirements.

A major goal of Phase 1 was the assessment of the feasibility of the transformational approach from visual models to mathematical models amenable to the automatic analysis of performance, performability and dependability, and to formal verification.

To establish the feasibility of the initial idea some types of analyses have been selected for starting and the associated transformations have been defined. In the second phase the intention is to consider other kind of analyses e.g. error propagation, performance, timeliness, for which the appropriate transformations will be worked out.

According to the workpackages defined in the Project Programme the following results have been achieved:

WP 1: Specification of the HIDE method

- **Task 1.1: Specification of modeling techniques**

In this task the basis for the development of UML-models amenable to quantitative analysis and formal verification has been provided. Mainly the following aspects have been investigated:

- Analysis of the UML-modeling paradigm:
  - Modeling of structures addresses the problems encountered in dependability modeling and analysis of complex systems.
  - Modeling of behavior deals with the process view that represents the system's concurrency and synchronization mechanisms. This view is captured mainly in Statecharts and Sequence Diagrams. For the purpose of formal verification it is necessary to aim at a precise formal semantics. UML does not yet provide such a semantics. Therefore, the informal semantics of UML-Statecharts is summarized and compared to the semantics of Harel-Statecharts. Then suitable subsets of UML-Statecharts exhibiting a precise semantics have been identified.
- Formal specification of an enriched subset of UML-modeling elements.
- Identification of a set of dependability attributes:

  Dependability attributes like safety and liveness which can be checked by formal verification are identified, and their representation is discussed. Also dependability attributes relevant to the quantitative analysis and their representations have been identified.

The details of the specification of modeling techniques are given in Deliverable 1.

- **Task 1.2: Specification of analysis and transformation techniques**

This task consists of three parts:

- Preliminary analysis of methods and tools

  Two categories of model types used for dependability analysis have been investigated: combinatorial models and state-space models, Markov models included. This latter approach has been widely accepted in the dependability community because of their powerful representative capabilities, and the relatively cheap solution technique.

The most important modeling paradigms for Markov processes are the Generalized Stochastic Petri Nets (GSPN). GSPNs can be seen as a standard language which is understood by the majority of the tools for automated evaluation of dependability attributes. Because of this portability, and of its availability and direct contact to the developers at FAU, within the HIDE project the tool PANDA (Petri Net Analysis and Design Assistant) has been selected.

- A set of transformations for the dependability model, the static model, and the dynamic model have been identified.
- Whenever possible, they have been formally defined.

Details of this specification can be found in Deliverable 2.

WP 2: Specification of the HIDE environment

The HIDE architecture has been defined in a form which can be used without major alterations for both phases of the project. The primary target of Phase 1 - the definition of a prototyping environment, considering the assurance of a high level of flexibility and of a good support for debugging - has been achieved.

The HIDE architecture is built up from the main components:

- **HIDE repository**: The main idea is the use of a database containing the actual UML model(s) and the functions executed on it. The advantages of this solution are:
  - Easy storage of persistent data
  - Same implementation platform for all functions
  - Platform independence
  - Rapid prototyping

  We implemented the HIDE repository on the base of the relational database system ORACLE.
- **UML CASE tool** as user-sided modeling platform. In the first phase of the project we used the INNOVATOR, provided by MID. This tool is a global CASE solution for analyzing and designing IT systems. The tool UmlNice, provided by INTECS is planned to be used in the second phase, too. All tools need extensions for the import and export of the UML models from or into the database, resp.
- **Verification and validation tools**: We used Spin (provided by Bell Labs) which supports the formal verification of distributed systems and PANDA (provided by FAU) a software tool for the analysis of timed Petri nets. These tools are “off-the-shelf” components and are not supposed to be altered in any way.

In addition, selected transformations of the UML models have been implemented.

Details of this specification are given in Deliverable 3.

WP 3: Feasibility study

- **Task 3.1: Assessment of analysis and transformation techniques**

This task dealt with qualitative and quantitative assessment criteria:

- Qualitative assessment criteria:
  - Adherence to the UML-modeling techniques

The basic input for the transformation from UML to timed Petri net models is represented by the standard *structural* UML diagrams, that is Use Cases, Classes, Objects and Deployment Diagrams. Since the UML specification does not cover the aspects related to the dependability of the system, these input diagrams are enriched by two extensions: one for identifying redundancy and the other for defining the dependability parameters.

The *dynamic* aspects of an UML model are captured in interaction diagrams (Sequence Diagrams), Statecharts and Activity Diagrams. We referred to the realization of this aspect by these diagrams as the Dynamic Model. We developed transformations for all of these diagrams without introducing new modeling elements.

The input for the translation from UML Statechart Diagrams to Kripke Structures is a subset of UML Statechart Diagrams and the translation itself is a way for providing such a notation with a formal operational semantics.

Therefore, adherence to the UML-modeling techniques is guaranteed.

- Semantic truthfulness of the transformations

  The transformation procedure from UML to timed Petri net models starts from the *structural* diagrams UML, and proceeds towards Petri net models. It is important to point out that the set of UML diagrams that forms the input of the transformation does not have a formal semantics.
  Thus, within the first step of the transformation, fault propagation and repair processes are described in a more precise way so that the intermediate model resulting from the UML diagrams becomes a proper input for the definition of a timed Petri net dependability model.
  Then a first step of the transformation is defined, which builds the intermediate model were each element is assigned an internal failure/repair process. Starting from the intermediate model, a second step of the transformation generates a timed Petri net model.

  For the transformation from *dynamic* UML diagrams to Stochastic Petri nets we made only a few restrictions with regard to the modeling power of some of these constructs to preserve the semantics of the UML-constructs.

  The translation from UML Statechart Diagrams to Kripke Structures is semantically sound. This is guaranteed by the formal proofs of some interesting properties of the intermediate model for the translation from UML Statechart Diagrams to Kripke Structures, namely Hierarchical Automata, and of their semantic model, including the correctness of such a model with respect to the informal definition of UML Statechart Diagrams.

- Availability of a complete description of selected transformations

  The transformations have been specified in a precise and unambiguous way. The algorithmic description of some steps of model generation have been provided.

  The complete formal definition of the translation from Hierarchical Automata to Kripke Structures is given. The translation from Statechart Diagrams to Hierarchical Automata is given informally but in a rigorous way.

• Quantitative assessment criteria:

It was out of the scope of the first phase of HIDE to provide a real quantitative assessment of the transformations. However some remarks related to the complexity of the resulting models and to a comparison with hand-made models for the same systems could be made.

The size of a final dependability Petri net model is proportional to the size of the intermediate model, whose size is in turns less or equal to that of the UML specification. The number of elements of the Petri net models is the same as the number of modeling elements of the original UML-models. Therefore, it goes without saying that 'the complexity of the automatically generated models is within an order of magnitude of the complexity of corresponding hand-made models'.

The definition of the translation from UML Statechart Diagrams to Kripke Structures is to be considered as a specification for the translation itself, to which any implementation of such a translation needs to conform. Being a specification, efficiency considerations do not make much sense.

The major consideration which can be done at this moment is that the translation from Statechart Diagrams to Kripke Structures is a way for providing a formal semantics for UML Statechart Diagrams within the HIDE environment. Availability of a formal semantics is essential for any formal verification.

Details of the assessment can be found in Deliverable 4.

- **Task 3.2: Development of demonstrator**

One of the main goals of Phase 1 was the development of a demonstrator to prove the feasibility of the HIDE ideas and to make a rough evaluation of the efficiency of the approach. The demonstrator consist of the following main parts:

- The HIDE core technology
- The transformation from the host UML tool’s repository to the HIDE database
- Production cell as demonstrator example

The Production Cell[1] has been selected as demonstrator example instead of the test environment for printed boards as announced in the PP, because this production cell is widely used as benchmark in modelling embedded systems.

The UML model of the production cell is of manageable size. The behaviour of the modelled system is described by a statechart which contains all the sub-statecharts of the components.

We performed some experiments with the transformed model which provided very useful experiences. The experiments employed the two transformations <Statecharts to Petri-nets> and < Sequence Diagrams to Petri-nets>, and the PN-tool PANDA . We also used formal methods to verify the specification of the production cell using the verification tool SPIN.

Details of the demonstrator can be found in Deliverable 5.

- **Task 3.3: Specification of the Pilot Application**

As a representative, real-life pilot application an Automatic Train Control (ATC) system has been chosen by the consortium. The specification reflects the current actual architecture of the system as already specified using informal notations or other formal methods as HOOD and SDL. In this respect the work performed is a kind of reverse engineering activity which tries to

1. C. Lewerentz and Th, Lindner, eds. Formal Development of Reactive Systems, vol. 891 of Lecture Notes in Computer Science. Springer-Verlag, 1995

exploit UML features to construct a more complete and understandable description of the system. This initial specification will be further elaborated and extended in the second phase of the project in order to make use and to assess the specific modeling techniques developed in HIDE.

In Deliverable 6 the system is described informally for introduction. There also a UML model of the static logical architecture of the system, a UML model of its principal dynamic behavior characteristics, and a UML model of the physical architecture are described.

1.2. Summary appraisal of the success

1.2.1. Assessment of the initial risks

The risks associated with the project originate in the fact, that a model transformation, correct in the sense of theoretical mathematics or computer science, does not necessarily provide a method efficient enough to be feasible in the terms of practical computer engineering.

However, our analysis of the transformations revealed that the model complexity does not increase considerably. In fact, for the selected transformations the complexity remains constant. Our experiments with the demonstrator showed that formal verification and quantitative analysis are feasible for non-trivial models.

1.2.2. Feasibility of the initial idea

Feasibility of the idea of translating Statechart Diagrams into state-transition diagrams for formal verification has been shown by the complete definition of the translation itself, and the complete proof of its correctness. Moreover, the well-foundedness of the basic ideas for such a translation is proved by the fact that a paper describing the translation and its theoretical framework has been accepted for publication in the proceedings of a major international conference in the field, namely the IFIP TC6/WG6.1 Third International Conference on Formal Methods for Open Object-Oriented Distributed Systems, FMOODS '99.

Feasibility of the idea of translating Statechart Diagrams into Generalized Stochastic Petri Nets has been shown by the definition of the translation and its implementation within the demonstrator.

1.2.3. Industrial relevance and possible benefits

The results of the HIDE project are of great interest and potentially relevant for a wide variety of industrial software application domains ranging from real time embedded systems to distributed systems including transaction based applications. This fact will be evident by the following statements of the both industrial partners of the consortium for technical and business applications.

Real time embedded systems constitute probably the principal domain for the exploitation of HIDE results, as can be easily seen from the requirements and the needs of the pilot application selected for the second phase of the project and more in general from the requirements that the organizations operating in this domain are expressing. However the exploitation of HIDE results in the real time domain can be considered as a starting point from which exploitation can spread to other domains. In general terms, real-time, embedded dependable systems are characterized by many peculiar factors that make them different from “normal” computer applications.

The software for this embedded systems is more difficult to construct than it is for desktop computers. Real time systems have all the difficulties of desktop applications plus many more. Usually non real time systems do not concern themselves with timeliness, robustness or safety - at least not nearly at the same extent of real time systems.

Real time systems encompass all devices with performance constraints. Hard deadliness are performance requirements that absolutely must be met. A missed deadline constitutes an erroneous computation and a system failure. In these systems, late data is bad data.

Virtually all real time embedded systems either monitor or control hardware or both. One of the problems that arises with environmental interaction is that the universe has the annoying habit of disregarding our opinions on how and when it ought to behave. External events are often not predictable. Systems must react to events when they occur not when it might be convenient. A train must stop immediately if something goes wrong along the track, an ECG monitor must alarm quickly following the cessation of cardiac activity.

Real time embedded system must often really optimize the usage of resources. Under Unix, if a developer needs a big array, he might just allocate space for 1,000,000 floats with little thought of the consequences. The embedded system developer cannot make these simplifying assumptions. Frequently real time developers must design and write software for hardware that does not exist yet. This creates real challenges since they cannot validate their understanding of how the hardware functions. Integration and validation testing become more difficult.

Embedded real time systems must often run continuously for long period of time. It would be awkward to have to reset your flight control computer because of a General Protection Fault while in the air. The same applies to Cardiac Pacemakers or unmanned space probes. Embedded system environments are often adverse and computer hostile. Solar storms generate strong radiation outside the atmosphere, cables that connect two train cars may be damaged or even cut. Even if the damage is not permanent, it is possible to corrupt memory storage, degrading performance or introducing system failure.

Apart from increased reliability concerns, software is finding its way even more frequently into safety systems, as medical devices, defence systems, nuclear and chemical plants, and vehicle control systems as aircrafts, spacecrafts, trains and even automobiles. Because of all these considerations it is quite clear the reason for the increasing interest of the industry towards the research of better ways to design and develop embedded real time systems. The Object Oriented technology claims to be a suitable answer to these needs. The primary advantages of Object Oriented development are:

- Consistency of model views
- Improved problem domain abstraction
- Improved stability in the presence of changes
- Improved model facilities for reuse
- Improved scalability
- Better support for reliability and safety concerns
- Inherent support for concurrency.

The Unified Modeling Language (UML) is an emerging standard visual notation for expressing the constructs and the relationships of complex systems. UML is more complete than other methods in its support for modeling complex systems. It is applicable in a wide range of application domains including business applications and processes, but as far as our specific interest is concerned, it is particularly suitable for modeling real time embedded systems. Its major features include:

- Object model
- Uses cases and scenarios
- Behavioral modeling with state charts
- Packaging of various kind of entities
- Support for multiple views of the system
- Representation of tasking and task interactions
- Model of physical topology
- Model of source code organization
- Support for object oriented patterns.

Business applications can profit from HIDE in the following areas:

- Quantitative dependability attributes:
  - Performance: Performance is always an important aspect of a software system. It is extremely useful to get some boundaries without an implementation of the system. If, for example, in an assurance company, the storage of the data into a database system is taken as a basis for the analyses one can decide whether the design is useful in an on-line operation or not.
  - Availability: A useful feature would be to give some forecasts of availability of the developed software system depending on some parameters that can be established easily. That would help a lot, for example to calculate costs for maintenance of the future system.
- Formal verification of dependability attributes:
  For a complete software system formal verification is almost impossible to afford. The formal specification of the system behavior is only done for critical aspects. For example, in banking software it could be important to be sure, that the software does not perform any transactions without an corresponding order. This feature can be useful, if the additional efforts that have to be taken are sustainable and are justified by the criticality of the involved software.
- Error propagation:
  The behavior of a system if some components fail, is of interest for every one who handles important information. It must be clear how errors propagate through a system to react to critical states. The analysis should include also the recognition of deadlocks.

The possibility of applying formal analysis and verification techniques starting from UML models is one key feature of the HIDE approach, which should allow to hide to the end-user most of the complexity that is inherent in formal notations and methods. The preliminary results emerging from the first phase of the project show the feasibility of this approach and hence encourage us to proceed along this line.

1.2.4. Achievement of the overall goals

The results provide a sound basis for a complete development of the HIDE environment. They show that our approach is feasible, and no major difficulties are expected, since we are able to perform, for a selected subset of modeling elements and transformations, the full set of tasks: modeling in UML, transforming the models, verifying and evaluating it.

1.2.5. Extension of the work

Based on the successful completion of the first phase the consortium considered an extension of the work into a second phase. This period should serve for the pre-prototyping of a commercial version of HIDE and should show the usefulness of the developed tool for real tasks by means of a representative pilot application.

With respect to the new funding situation it is intended to submit a proposal for the Fifth Framework.

1.3. Management aspects

For the first phase of the project Wolfgang Hohl from FAU served as Program Coordinator.

Before the project started on June 15th, the partners of the consortium met for the preparation of the proposal and of the Project Programme in Erlangen (December 1997).

During the funding phase the partners have been in permanent contacts to discuss the current work, solve arising problems, and plan the second phase: They met several times in smaller groups or individual visits, and three times for plenary sessions in:

- Munich (June 22-23): Kick-off meeting
- Pisa (September 23-25): Progress meeting; at this meeting a talk on an Automatic Train Control (ATC) system has been presented by INTECS. The ATC should serve as pilot application for experimenting and assessing the modeling and analysis techniques developed in HIDE in the second phase.
- Budapest (November 25-27): Final meeting to prepare the Final Report and to discuss a possible extension of the work.

In addition to the funding payment considerable resources have been made available by the partner institutions. Research was done not only by funded people but also by the staff and Ph.D. students.

Not only the PANDA tool developed by FAU but also the tools of the industrial partners, INNOVATOR by MID and UmlNice by INTECS, was made available free of charge within the consortium. As planned in the Project Programme a complete version of UmlNice will be made available to all the consortium partners in the second phase of the project.

Contacts with candidates for associate partnership from the field of European tool providers and pilot users in order to extend HIDE with further aspects of system design, like safety or security have been established.

The internal project reports have been made available on the official HIDE homepage, URL <https://asterix.mit.bme.hu/mainpage.html>, maintained by TUB. As a first scientific output of the project two papers have been accepted/published at international conferences.

2. Criteria

Specific assessment criteria are listed in Deliverable 4.

1. Adherence to UML:
   - Evaluation results have tangible value to the modeler working with UML
   - Input/output of evaluations clearly correspond to UML modeling elements

   Both aspects have been assessed in the specifications for modeling and transformations.
2. Feasibility of technical architecture:
   - The feasibility of an interfacing between CASE tools and analysis tools has been proven by the demonstrator implementation
   - The openness of the architecture towards the host environment and the requirements on this UML toolset is guaranteed by the core of the HIDE architecture.
3. Ease of understanding:
   The mathematical details are largely hidden for the final user, making the HIDE toolset usable by users without an expertise in abstract mathematics.
4. Reliability of properties predictions:
   The estimation and the prediction of the system properties performed with HIDE techniques and tools have an acceptable level of confidence.
5. Efficiency of transformations:
   The somewhat larger model size and run-time requirements of the automatically derived models does not strongly limit the target field of applications in comparison with the manually composed models.

During the first phase of the HIDE project the formal definition of the translation from UML to Kripke Structures has been fully developed (see Deliverable 2) and its correctness has been formally proven (see Deliverable 4).

Such a definition is to be considered as a specification for the translation itself, to which any implementation of such a translation needs to conform.

No definite **quantitative** assessment which refers to implementation issues, like memory usage or execution time can be done at this stage of development of the work.

The **qualitative** criteria of the translation have been assessed (compare Deliverable 4). They are listed below :

QL1) Adherence to the UML-modeling techniques

QL2) Semantic truthfulness

QL3) Availability of the formal definition of the translation.

3. Deliverables

The following deliverables have been completed:

Deliverable 1: Modeling

- Report on the *Specification of Modeling Techniques*

Deliverable 2: Transformations

- Report on the *Specification of Analysis and Transformation Techniques*

Deliverable 3: HIDE environment

- Report on the *Specification of the HIDE Environment*

Deliverable 4: Transformation assessment

- Report on the *Assessment of Analysis and Transformation Techniques*

Deliverable 5: Demonstrator

- Software and Report on the *Demonstrator*

Deliverable 6: Pilot application

- Report on the *Specification of a Pilot Application*

Deliverable 7: Final report