

# IDERS: An Integrated Environment for the Development of Hard Real-Time Systems \*

Alejandro Alonso  
ETSI de Telecomunicación  
Ciudad Universitaria s.n.  
E-28040 Madrid, Spain  
e-mail: aalonso@dit.upm.es

Luciano Baresi  
Politecnico di Milano  
Piazza Leonardo da Vinci 32  
I-20133 Milano, Italy  
e-mail: baresi@elet.polimi.it

Hanne Christensen  
The Institute of Applied Computer Science  
Forskerparken 10  
DK-5230 Odense, Denmark  
e-mail: hanne@ifad.dk

Marko Heikkinen  
VTT Electronics  
Kaitoväylä 1, Linnanmaa  
FIN-90571 Oulu, Finland  
e-mail: Marko.Heikkinen@vtt.fi

## Abstract

*IDERS is a new generation environment for developing real-time critical systems. It integrates specification, design and code within a single framework, provides support for incremental prototyping and allows early validation through testing and animation. The system is based on a kernel that eliminates ambiguities and supplies dynamic semantic checks. Customization facilities allow to tailor the environment to specific notations, preserving the benefits of the formal kernel.*

*The software process is modeled by a process-centered software engineering environment that guarantees the complete visibility of both the development process and the evolving products. The IDERS project supplies a specific process model inspired by the Boehm's spiral life cycle model.*

## 1 Introduction

The development of embedded real-time systems undergoes several stages, that include requirements specification, design, and implementation, and involves many orthogonal activities, such as testing, verification and validation. The currently available tools and environments support only some of the development phases, without providing a coherent development framework.

The IDERS project, described in this paper, provides an integrated environment that supports the whole software life cycle of real-time systems. IDERS proposes an original strategy based on complete visibility of the evolving product and of the development process. Visibility of the evolving product is achieved with a set of tools and methodologies that strongly integrate all development stages from requirements specification to final code. The development framework allows components of the final system to be incrementally integrated to build early prototypes, despite their current maturity: prototypes can be produced incrementally from a system description partially given in terms of requirements specification, partially in terms of design specification, and partially in terms of final code. Incremental prototypes allow early validation through testing and animation. Complete process visibility is provided by the enactment of a process description. The integration between the development process and the evolving product is guaranteed through the support of a process-centered software engineering environment (PSEE): SPADE [1].

This paper describes the main novelty of the IDERS environment. The original architecture integrating process description and enactment with distributed support for requirements and design specification, code testing and incremental animation is described in Section 2. The IDERS process model is presented in Section 3. Methodologies and tools for supporting requirements and design specifications, and adaptation with host-based animation are described in Section 4. The original solution for allowing customizability is described in Section 5. The development plans for the near future and the applications currently used as benchmarks are outlined in Section 6.

---

\*The work presented in this paper is partially supported by the European Communities under the ESPRIT programme, project no. EP8593. The IDERS consortium comprises: IFAD - The Institute of Applied Computer Science (Denmark), VTT Electronics (Finland), Politecnico di Milano (Italy), MARI - Computer System Limited (United Kingdom), Universidad Politécnica de Madrid (Spain), ALENIA - un'azienda Finmeccanica (Italy), Rautaruukki New Technologies (Finland), Lattice limited (United Kingdom).

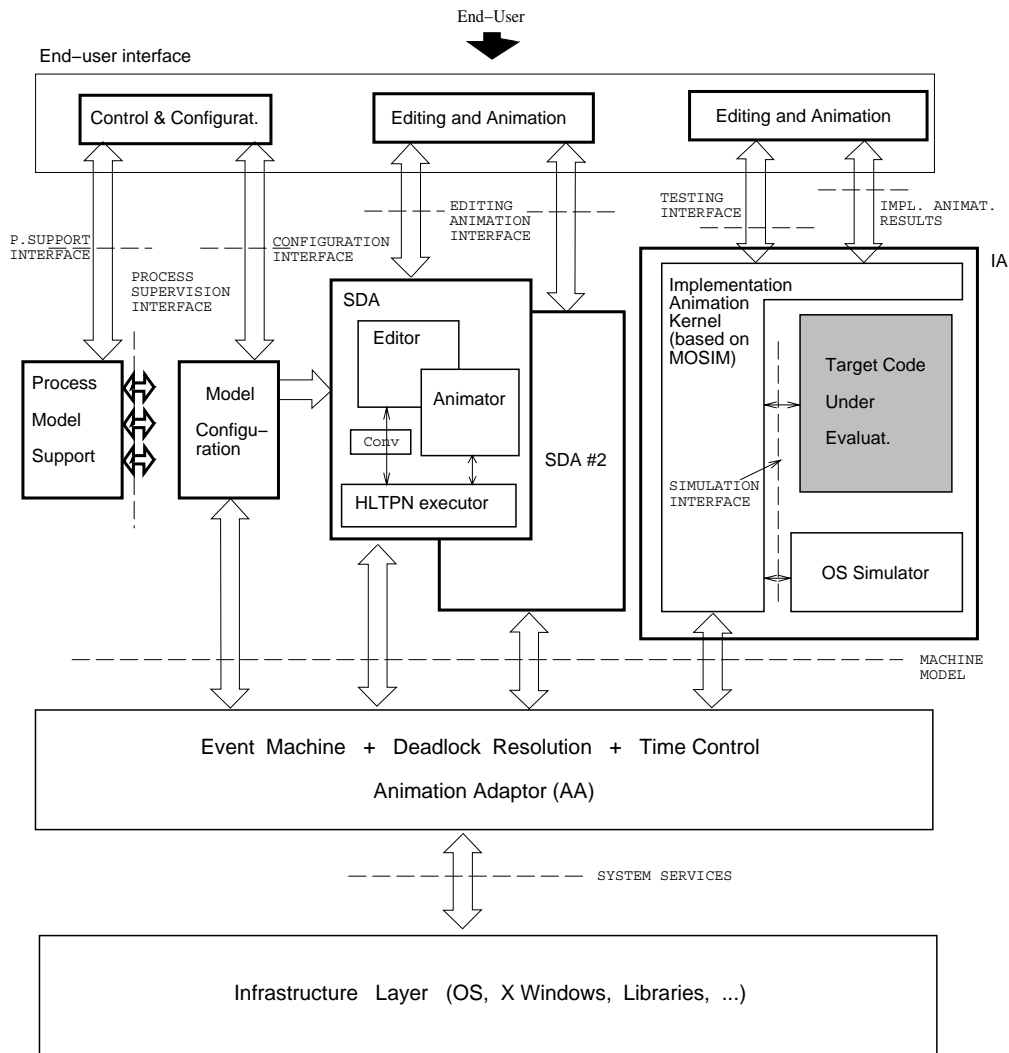


Figure 1: The overall architecture of the IDERS environment

## 2 IDERS Architecture

Several issues have been considered in designing the architecture, the most important ones are:

- Even if the IDERS toolset is a meta-environment that can be customized to cope with the notations and the development standards of any specific organization, the architecture has been thought considering the abstraction levels provided by SA-SD/RT methodology [14]. This approach was chosen just to focus on a specific methodology, being aware that its basis are common to other methodologies. Basically SA-SD/RT requires the following models:
  - *Specification Models (SM)*: They define the functional requirements of the system under development. Based on the hypothesis that system resources are unlimited, they mainly describe the environment with which the system will interact, and the system responds to some meaningful events.
  - *Design Models (DM)*: They specify the system at a lower abstraction level taking into account system resources. It is here that the system is partitioned into subsystems and the interfaces between the different components are specified.
  - *Implementation Models (IM)*: These models are simply the final system code. In addition, it could be augmented with special code (drivers and stubs) for simulating some aspects of the target systems, for instance I/O devices.
- Openness, configurability, and multifaceted integration (visual, data, control and process integration).

- Support for the IDERS process model, i.e., tools for specifying the process model and controlling its evolution.

The complexity of the IDERS toolset makes advisable the development of different views of the architecture: Figure 1 gives a general view, where special emphasis is put on the components and their interrelations, while Figure 2 and Figure 3 provide particular viewpoints perceived by the different users of the system.

The main modules identified in Figure 1 are:

- *The Process Model Support (PMS)*, responsible for ensuring complete visibility and control of the development process model. It is based on SPADE [1], whose core components are developed within a different ESPRIT project<sup>1</sup>. The IDERS Project enhances SPADE by providing additional tools for controlling the process: agenda and monitor.
- *The Model Configuration (MC)* lets the users customize the environment for animating a system description. It must be specified: the submodels that are part of the system, where they are going to be located, in case of distributed animation, and the test cases used as inputs.
- *The Specification & Design Animation (SDA)*, that allows the definition and animation of (sub)models by means of the requirements specification and design notation for which the IDERS environment has been customized. It translates both the (sub)models to be animated into high-level timed Petri nets [5], which are the internal notation of the toolset, and the execution events, at the Petri nets level, into significant actions according to the end-user notation.
- *The Implementation Animation (IA)*. The main purpose of this toolset, whose kernel component is based on MOSIM [8], is the execution and visualization of Implementation Models (IMs) in a host environment that simulates the target environment.
- *The Animation Adaptor (AA)* provides the basis for the integrated animation of the heterogeneous models and thus the consistent execution of the whole system.

Figures 1, 2 and 3 identify three different kinds of users:

- end-users: the experts of the application, e.g., the expert of radar controllers. They interact with IDERS through a set of specific notations, suitably used for the developed application. They use the IDERS environment for requirements and design specifications, animation, incremental prototyping and host based code simulations. The development process is guided according to the

chosen process model by the SPADE process enactment engine.

- super-users: the experts of the customization notations. They interact with the IDERS customization components, as shown in Figure 2, to formally define specification and design notations required by the end-users. They define concrete and abstract syntax, semantics, and animation rules. Graphical customization is based on graph grammars: the super-users define connectivity rules (i.e., abstract syntax) and the high-level timed Petri net corresponding to an end-user specification by means of suitable graph grammars. Animation rules allow the execution of the “semantic net” to be shown in the chosen end-user notation.
- process-managers: the experts of the PSEE SPADE. They define the process model that drives the software development. The designed process model is enacted to drive the design process. Their viewpoint within the overall architecture is drawn in Figure 3.

### 3 The IDERS Process Model

The IDERS Project provides a specific process model tailored on the needs of the demonstrators and based on the results of the ESPRIT II Project EP5570 IPTES [12] and on the Boehm’s spiral life cycle model [3] referring to the U.S. Air Force/IBM Software First Life Cycle (SFLC) Model [2], already proven well suited for the development of real-time systems. In terms of the IEEE Standard for Developing Software Life Cycle Processes, the focus is on the development process.

The SFLC model consists of four main phases in which: the system and users’ requirements are well understood, a full capability system prototype is evolved to address development risks early and to deeply evaluate and refine system capabilities, the prototype is converted into a productized system and finally the productized system is operated and maintained at the customer site.

The major improvements in IDERS with respect to the SFLC, concern the explicit definition of process control and behavior, incomplete or missing in the original description, the management of multiple instances of the same activity and the introduction of explicit integration activities earlier in the development process.

An important role of the IDERS process model is to provide guidelines how to utilize the various IDERS tools: the customization toolset, the SDA and the IA in the development of embedded software:

- the customization toolset is mainly used during the preliminary activities for selecting and customizing the development environment.
- The SDA is employed in the prototyping-intensive phases to allow incremental prototyping and to increase process visibility by graphical animation.

<sup>1</sup>The ESPRIT Project EP6115 GOODSTEP: General Object-Oriented Databases for SoftWare Processes.

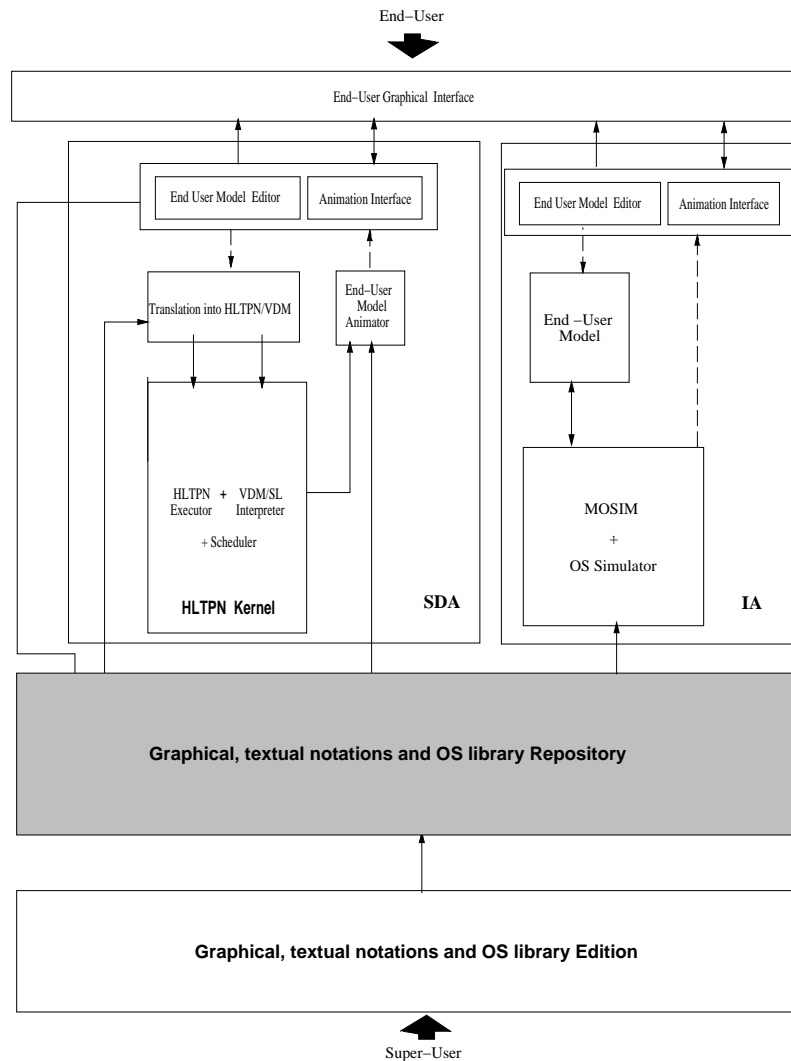


Figure 2: The super-user's point of view within the IDERS architecture

- The IA provides host-based testing and OS simulation that are to be used both during the prototyping phases, in parallel with the SDA, and during the integration of the various components.

Besides this, the PMS allows process managers to adapt the IDERS environment to cope with the needs of specific development processes.

Finally as the whole model is quite big, it is not feasible to define the entire model into an enactable form within the IDERS framework. Instead portions, directly related to the IA and the SDA usage, will be refined to become enactable.

#### 4 Specification, Design and Implementation support

IDERS provides the end-users with notations for requirements and design specification. There will be

an initial set of notations supplied with the first prototype. Additional notations will be later added using the customization facilities described in the next section. The first release of the IDERS SDA will support the notations proposed by Hatley and Pirbhai (H&P) [7] augmented with *minic*, a dialect of *C*, for expressing textual aspects. The requirements notation strictly adheres to H&P, while the design notation integrates the original proposal with the POSIX standard [9] as to the elements not fully specified.

Requirements and design described by the end-user in the H&P notations are internally represented by means of high-level timed Petri nets. Consequently, one of the side results of the IDERS project is to give complete formal semantics to the H&P notations by means of high-level timed Petri nets.

The “semantic” high-level net underlying the end-user specifications provides many advantages:

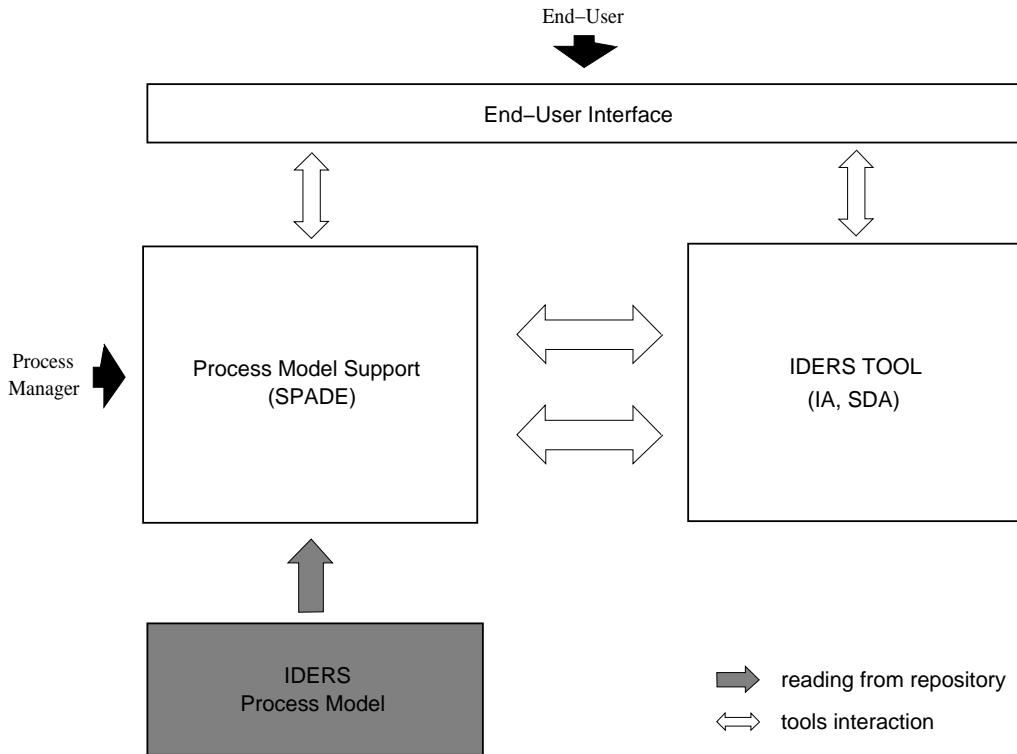


Figure 3: The process-manager's point of view within the IDERS architecture

- It adds formality to a semi-formal notation. The IDERS environment provides not only syntactic and static semantics checks, but also feedback on the dynamic semantics of the specifications: all the significant events at the Petri net level are presented back to end-users according to their own notations.
- It provides the basic support for execution and animation of specifications. The IDERS environment fully supports the animation of the graphic notations and the execution of the textual annotations. Execution and animation are strictly inter-related: graphic animation is driven by the results of the execution of the textual annotations of the graphic components. For instance, referring to the IDERS requirements notation, the execution of a process annotation, suitably expressed in terms of Petri nets, induces process highlighting, i.e., the animation event associated with executing processes. The resulting animation tools greatly enhance the possibilities of revealing errors early in the specification and design phases.
- It provides the basic support for incremental prototyping and code integration. Since all the stages of development are internally represented in terms of high-level timed Petri nets, systems descriptions can be animated and verified regardless the stage of development of each compo-

nent. IDERS can animate a system specification comprising detailed design of part of it and initial specifications of the requirements of other parts. Suitable communication and synchronization mechanisms between high-level timed Petri nets and target code allow early integration of code in a system comprising components not yet fully implemented.

- It finally provides a homogeneous framework for supporting distributed design: suitable distribution mechanisms allow a single specification to be geographically distributed and simulated.

In general, the executable models will be heterogeneous comprising parts defined both by the SDA and by the IA, and physically distributed, i.e., different sub-models located and executed on different machines.

End-users define SDA models by interacting with a graphical editor<sup>2</sup> customized, as explained in the next section, for the chosen notation. Each graphical symbol or textual annotation added to the model is translated into an abstract (internal) form, subsequently used both as basis to build the Petri net model and to

<sup>2</sup>Within the IDERS project, it has been chosen Software through Picture (StP) due to its leadership in the CASE tools market. However, other tools matching a minimal set of requirements can be chosen.

map execution events at the kernel level back to the end-user interface.

IA models handle source code and are translated referring to specific programming language rules, OS primitives and some basic libraries. From a visualization viewpoint, the key element can be either the original code, pointing out the current instruction under execution and the valued variables, or a higher-level view in terms of modular description, where interesting information becomes the OS and I/O events among the modules.

The integrated animation between the SDA and IA models is provided by the AA, that is in charge of ensuring consistent animations of the whole system.

## 5 Customizability

The IDERS environment provides meta-tools for adapting itself to the specific needs of the end-user and of the final application. Both graphic and textual aspects of the end-user notations are customizable. Graphics customization is based on graph-grammars theory [6]: both syntax and semantics aspects are expressed by suitable graph grammars that are interpreted to provide editing support for the customized notation and to generate the kernel model. The textual customization encapsulates syntax, static and dynamic semantics. The customization of static and dynamic semantics is specified and described using VDM-SL [10], that can be either executed through the VDM-SL<sub>to</sub>C++ Code Generator or interpreted by the VDM-SL interpreter [4].

As already outlined in Section 2, in customizing the environment for an end-user notation, super-users, as far as graphical aspects are concerned, must define:

- The concrete syntax (appearance) of the symbols, i.e., all the shapes used within the notation. Of course, the actions needed to perform this task highly depend on the chosen end-user interface.
- The abstract syntax: a finite set of graph-grammar productions defining the internal representations of the notation elements and the “legal” actions to manage them.
- The semantics: a finite set of productions, one for each production belonging to the abstract syntax, that, given an action at the abstract level, teach the system how to update the kernel model.
- The animation rules that translate the firing of transitions or the marking of the kernel model into logical events, belonging to the abstract level, and then into concrete graphical actions. For instance, it can be programmed that the firing of a certain “kind” of transitions, meaning starting the execution of processes, could be seen by the end-users as the placement of a “concrete” token inside the circle representing the process.

While, as to the textual part they must address:

- The concrete syntax, defining language keywords and constructs.

- The static semantics, defining the type correctness and module compatibility.
- The dynamic semantics.

Other aspects being customizable by super-users are:

- The static semantic checks between the graphical and the textual components of the customized notation.
- The scheduling policies employed to execute the end-users’ models.
- The textual part of the high-level timed Petri net, referred to as transition specifications.

Customizability provides several advantages:

- it supports the evolution of the environment according to the evolution of specification notations. Unlike most existing CASE tools, the IDERS environment can be easily adapted to new forthcoming notations, guaranteeing a wide stability of the environment in the future.
- it allows IDERS to be tailored to specific applications. Many application do not have a market wide enough for justifying the development of supporting CASE tools. Presently, the developers has only two choices: either developing the application with little support of CASE tools or using different notations supported by exiting CASE tools. In both cases the results are not too encouraging: in the first case most benefits derived from the use of automated CASE tools are lost, in the second case, the specification can contain errors due to the use of inappropriate notations. The customization support of the IDERS environment opens a third alternative: customizing the environment for the desired notation with a reasonable effort.
- it support experimentations of new notations, while existing CASE tools tend to restrict the notations to the few that find good support limiting the creativity of advanced development teams, and thus the competitiveness of the specification and design techniques in the future.

Customization facilities are tested within the IDERS project by defining the full H&P notations.

## 6 Conclusions

IDERS is a two year ESPRIT project entering its second year. Most specification and design activities have reached their conclusions and the main implementation activities already started:

- The specification and design of the SDA has been completed, and the implementation phase has initiated. Customization of the SDA according to the H&P notation augmented with the *minic* is in progress. A first internal release is scheduled by the end of July.

- The first basic IDERS process model has been defined. The specification of the new tools has been completed and the development of prototypes initiated.
- Functional requirements of the AA have been defined, and early prototypes developed.
- The specification of the IA has been completed, some of its components have been developed and the integration IA-AA has started.

As soon as the SDA will be released, it will be proved and demonstrated on the logical and physical models of the Reply Processor and Channel Management System (RPCMS) [11], the key software component of an advanced secondary surveillance radar system developed by Alenia. The RPCMS is required to be able to manage aircrafts equipped with both traditional and Mode-S transponders. Mode-S provided aircrafts own a unique address that let interrogations be directed to single aircrafts. The basic surveillance procedure is split into two parts: in the first phase the radar queries all the aircrafts in the current sector, that reply with their own data, while in the second phase it interrogates selectively the Mode-S equipped aircrafts, i.e., the only ones owing a unique identifier, to ensure and improve surveillance. It is, therefore, mandatory that all the aircrafts within a given sector are queried and their replies received back before the radar changes sector, otherwise significant information would be lost.

The IA is used in the development and enhancement of the Smartvis [13] system developed by Rautaruukki New Technology. Smartvis is a surface inspection system for flat steel products based on high-resolution line scan imaging, a special illumination system, advanced digital filtering methods, rule-based defect classification and an automated learning and tuning support system. The main focus of the IA usage is in making the customization of the defect classification portions of Smartvis easier. This is achieved using the host-based simulation features of the IA that include real-time operating system and hardware simulation facilities.

## References

- [1] S. Bandinelli, A. Fuggetta, and C. Ghezzi. *Software Process Model Evolution in the SPADE Environment*. *IEEE Transactions on Software Engineering*, 19(12):1128–1144, December 1993.
- [2] M. Blumberg and M.C. Ward. *Software-First Life Cycle Final Definition for the Software Technology for Adaptable Reliable Systems. Document number 1240-001. IBM Systems Integration Division, 800 North Frederic Avenue, Gaithersburg, Maryland, page 82, January 1990.*
- [3] B. Boehm. *A Spiral Model of Software Development and Enhancement*. *IEEE Computer*, 21(5):61–72, 1988.
- [4] R. Elmström, P. G. Larsen and P.B. Lassen. *The IFAD VDM-SL Toolbox: A Practical Approach to Formal Specifications*. *ACM Sigplan Notices*, 29(9):77–80, September 1994.
- [5] C. Ghezzi, M. Felder and M. Pezzé. *HLTPN Kernel Model*. Technical report, Politecnico di Milano, July 1992. IPTES Doc.id.: IPTES-PDM-6-V3.0.
- [6] H. Göttler. *Graph Grammars, A new Paradigm for Implementing Visual Languages*. In *Proceedings of ESEC89-Second European Software Engineering Conference*, pages 336–350. Springer-Verlag, 1989. Lecture Notes in Computer Science, n. 387.
- [7] D.J. Hatley and I.A. Pirbhai. *Strategies for Real-Time System Specification*. Dorset House, New York, 1987.
- [8] H. Honka. *A Simulation-Based Approach to Testing Embedded Software*. VTT Publications 124, Technical Research Center of Finland, Computer Technology Laboratory, P.O. Box 201, FIN-90571, Finland, 1992.
- [9] IEEE. *Real-Time extensions to POSIX. IEEE Standard P1003.1b*. IEEE, March 1993.
- [10] P.B. Lassen, P.G. Larsen and K. de Bruin. *The IFAD VDM-SL Language*. Technical Report IFAD-VDM-1, IFAD, The Institute of Applied Computer Science, Forskerparken 10, DK-5230 Odense, Denmark, January 1994.
- [11] A. Perrotta, G. Massara, G. Dipoppa, and R. Bove. *A Logical Model of the RPCM System*. Technical report, Alenia, January 1995. IDERS Doc.id. : IDERS-ALENIA-16.
- [12] P. Pulli, M. Heikkinen, R. Lintulampi, J. Matero, and T. Piironen. *IPTES Spiral Model Guidelines*. Technical report, Technical Research Center of Finland, Computer Technology Laboratory, P.O. Box 201, FIN-90571 Oulu, Finland, September 1993. IPTES Doc.id. : IPTES-VTT-34.
- [13] Rautaruukki New Technology. *SMARTVIS Surface Inspection System for Continuous and Real-Time Inspection of Rolled Metal Strips*. Technical report, Rautaruukki New Technology, Teknologiantie 2, FIN-90570 Oulu, Finland, 1994.
- [14] P.T. Ward and S.J. Mellor. *Structured Development for Real-Time Systems*, volume 1-3. Yourdon Press, 1986.