



Evolution Management and Process for
Real-Time Embedded Software Systems

Essentials and Requisites for the Management of Evolution – A Formal Base for Evolution

D1 part 4
Edited by David Sellier

25 April 2002
Version 1.0
Status final

Public Version



I T E A

INFORMATION TECHNOLOGY

FOR EUROPEAN ADVANCEMENT

This document is part of the work of the EUREKA Σ! 2023 – ITEA 00103 project EMPRESS.
Copyright © 2002-2003 EMPRESS consortium.

Authors/Partners:

Partner	Author	Email
European Software Institute	David Sellier	David.sellier@esi.es
Cefriel	Luigi Lavazza	lavazza@cefriel.it
Jabil Hasselt	Linde Loomans	Linde_loomans@jabil.com

Document History:

Date	Version	Editor	Description
10 12 2002	1.0	Stefan Van Baelen	Public version based on internal version 1.0

Filename: D1.4_v1.0_Public_Version.doc

Abstract

Reasoning about the change implies the investigation of various aspects of the changes. Areas such as the product description notation and methodology, methodology for the change process description, techniques for managing changes within projects as well as an analysis of the tools supporting the change modelling or analysis should be investigated in order to provide a global idea of what is actually investigated and used in the “real world”.

The analysis summarises research activities, and technologies available that can deal with the needs and requirements introduced with the change during the development and the operation of embedded system. The analysis gives as a result a description of the techniques, methods, technologies that are currently used in the industry.

This analysis not only takes into account the projects dealing with the change modelling and change process management (that can become a valuable source of information for several research issues related to the change activities) but also the industrial needs and the research questions about product evolution.

Keywords:

Change management, change process modelling, change process techniques, product modelling, and domain specific language.

Table of Contents

Abstract	3
Table of Contents	4
1 Introduction	6
2 Topic 1: Reasoning about the Changed Product	6
2.1 State-of-the-Art	7
2.1.1 Issues Concerning Representing and Reasoning about the Changed Product.....	7
2.1.2 Extensions of UML for Dealing with Time Issues.....	8
2.1.3 Formalization of UML	9
2.1.4 Model Checking.....	9
2.1.5 Modelling Techniques for Product Families	10
2.2 State-of-the-Practice	10
2.2.1 Modelling Techniques and Formal Verification	10
2.2.2 Modelling Techniques for Product Families	11
2.3 Public Projects	11
2.4 Industry Needs	12
2.5 Research Questions	12
2.5.1 Main Research Directions	12
2.5.2 Behavioural Subtyping	14
2.5.3 Modelling Techniques for Product Families	15
3 Topic 2: Reasoning about the Change Process	15
3.1 State-of-the-Art	15
3.1.1 Change Process Modeling – Notation – Representation	15
3.2 State of the Practice.....	17
3.2.1 Change Process Modelling – Notation – Representation	17
3.2.2 Process Description Configuration Management.....	17
3.3 Public Projects	18
3.4 Industry Needs	18
3.4.1 Versioning Management and Change Management.....	18

3.4.2	Product Family Evolution Process	18
3.5	Research Questions	19
3.5.1	Change Processes	19
4	Topic 3: Techniques and Tools to Support the Change Process	19
4.1	State-of-the-Art	19
4.1.1	Change Estimation	19
4.2	State-of-the-Practice	19
4.2.1	Tool Support for Product Family Change Estimation.....	19
4.2.2	Current Process and Methods	20
4.2.3	Tools Supporting Change Management Process	20
4.3	Public Projects	21
4.4	Industry Needs	21
4.4.1	Improvements to Current Processes	21
4.4.2	Configuration and Change Management Tools	21
4.5	Research Questions	22
4.5.1	Product Family Change Management Information.....	22
5	Conclusion	22
	Literature.....	22

1 Introduction

WP4 deals with the precise representation of the various aspects of changes: the product to be changed, the required change, how the change is performed (i.e., the change process). The goal of the WP4 is to provide developers with the means to reason about the evolution of a product on the basis of models. The information provided by these models can be very different: change specifications can help verify whether the proposed changes maintain/achieve the desired properties; empirical knowledge of the costs of change activities can help estimate the cost of a change; etc.

This document contains a survey of state of the art and state of the practice in the domains of the change management on embedded systems. These reasoning have been divided into three topics that are:

- Reasoning about the changed product: this part concerns the convenient notation and methodology for describing product and required changes.
- Reasoning about the change process: This part embeds the reasoning about the notation, methodology or model definition for describing the change process and the techniques or methods employed in effective change processes.
- Techniques and tools to support the change process: this section is an analysis of the current tools used in order to manage various aspects of the change.

In order to give a complete view of what is doing in the real word about change process and product evolution techniques, each topic has been divided into five parts:

- The state of the art: this section describes the techniques, methods and technology currently investigated and available on the market to manage the different aspects of the changes.
- The state of the practice: this section describes the techniques, methods, methodology and model currently used by the industries.
- The public projects: this part enumerates the different projects related to the different topics of the WP4.
- The industry needs: this section describes the industry needs related to the change modelling, change process management and other aspects of the change.
- The research questions: The research question include open question concerning the product evolution.

2 Topic 1: Reasoning about the Changed Product

In the first task (T4.1) of the WP we shall propose a convenient notation and methodology for the precise description of the product and of the required changes. The goal is to provide a description of the system artefacts, the properties of the system that are subject to user requirements, and the required changes at various levels (requirements, analysis, design). Such a description will serve as a reliable basis for reasoning about proposed/possible changes.

This chapter establishes a set of topics to be explored in order to provide a sound basis for the activities mentioned above.

2.1 State-of-the-Art

2.1.1 Issues Concerning Representing and Reasoning about the Changed Product

This section illustrates the issues concerning the representation of changes, the way to reason about changes and the way to implement them.

2.1.1.1 Modelling Techniques

We need notations to represent the product. As long as the product is software, we need to represent typical software artefacts. The idea of developing software according to a model-based approach is gaining acceptance, and has been already employed in the DESS project. Here we have to explore which of these notations (and related methodologies and tools) are interesting for the purposes of EMPRESS.

In particular we have to consider notations that apply to

- object-oriented software
- component-based software
- real-time software

Possible directions for the exploration are:

- What to reuse from the DESS experience
- Formal notations available
- UML 2.0
- Other UML extensions/adaptations/formalizations

2.1.1.2 Components and Composability

Component-based development has an important role in the EMPRESS project. It is therefore required that we explore components and composability or component interaction issues such as compatibility, replaceability or adaptation. We need to explore:

- notations,
- semantics,
- support for developers (tools, methods, etc.), specially concerning specification, analysis, verification, design,

2.1.1.3 Implementation Issues

As long as we have to deal with existing software products it is of vital importance to be able to understand and describe how such products are built. We should therefore explore the features of the implementation-oriented facilities that are available to support the development of real-time (object-oriented) software:

- Middleware
- Architectures

- Available COTS (Commercial-Of-The-Shelf) components and the environments where they can be used.

2.1.1.4 Change

Of course change is a central issue in the EMPRESS project. At this stage, however, it is probably sufficient to explore the way changes can be defined and described. Nevertheless, additional information on change management is welcome.

The main topic could be:

- Taxonomies of changes

2.1.1.5 Behavioural Subtyping

Behavioural subtyping is a much more research-oriented topic. It deals with using types to guarantee the behavioural compatibility of components that are subtypes (subclasses) of the “ideal” components, which provide the specified behaviour.

2.1.1.6 Product Families

The control and monitoring software in an embedded system should be customised in line with the features of the physical system where it is integrated. There are some types of products like the cars, or home appliances, or... that are very similar and therefore their software is also very similar. The software product families applies the same concepts of the product lines to the software domain. This is, based on the user requirements the product line generate a final product that complies with the needs of the user.

A software product family consists on a set of artefacts that make possible to obtain the right product based on the user requirements

2.1.1.7 Trends

This section addresses all the issues that do not fit easily in the above sections, but are nevertheless interesting for our purposes. For instance, Model Driven Architecture (MDA) is interesting because OMG appears to be committed to promote this technique. As long as OMG has a great power of influencing the industrial practice

2.1.2 Extensions of UML for Dealing with Time Issues

Several efforts have been done to employ UML in the development of real-time software.

Douglass has shown how several features of UML can be exploited in the specification and design of real-time software [Dou98]. However, he has not solved the problems connected with the lack of formal semantics of UML. Moreover, he has not tackled difficult problems which –like the Generalized Railroad Crossing (see [Lav01])– cannot be modelled by means of plain UML.

Selic et al. have proposed the Real-Time Object-Oriented Modelling (ROOM) language, which has been merged with standard UML to form a proposal of ‘UML for real-time’ [Sel99]. This language is being used in several tools, like Rose for Real-Time and Anylogic. ROOM favours the design and coding activities, at the expenses of the specification phase. The language provides rich information on the structure of the system thanks to the facilities provided to describe active objects and their communications, and tools are available to translate the models into executable code which behaves like the models. The limits of ROOM are in the

lack of constructs to describe non-trivial time constraints, as well as in the lack of formal semantics (semantics is hardwired in the ROOM virtual machine). As a consequence, a ROOM model cannot be analysed in order to prove whether the model has desirable properties such as safety.

2.1.3 Formalization of UML

Other efforts are being spent in order to improve the formality of UML.

cTLA (compositional Temporal Logic of Actions) adds temporal logic to existing UML diagrams [Gra00]. cTLA+ [Her97] is a compositional specification and verification technique based on Leslie Lamport's Temporal Logic of Actions TLA [Lam99]. cTLA+ supports modular process type definitions and the composition of processes into systems. Processes can model components of an implementation. Moreover, they can represent modular logical constraints.

There are many other approaches aiming at a formalization of UML, all of these make UML more usable in critical, embedded and real time systems.

Two main efforts that are now converging in UML 2.0 (still in draft version [UML01a][UML01b]) are ROOM (see above) and Catalysis [Sou98].

The approach in Catalysis is to use OCL (Object Constraint Language see [OMG01]) constraints to eliminate most of the ambiguities that UML leaves unaccounted, i.e., OCL is used to formally define the diagrams in UML. Catalysis also introduces a notation to model components in a similar yet more accurate way than ROOM.

UML 2.0 adopts most of the notations of ROOM and some of Catalysis ones (in particular the usage of OCL to define diagrams).

Catalysis also inspired the Precise UML (pUML) project, which eventually evolved in the 2U (Unambiguous UML) consortium [2UC02]. The approach in pUML is to formally express the UML Metamodel through OCL, with special attention dedicated to conformance and compatibility between different UML views, as well as to refinement.

The pUML project has developed a Meta-Modeling Framework (MMF) [Cla01] as a replacement of the original UML Meta Model, in order to eliminate many of the current ambiguities and deficiencies, and to make UML more modular and extensible. OMG is considering to fully adopt this approach in UML 2.0 [UML01a][UML01b].

Other approaches propose the usage of UML combined with a formal language. For example, in [Bru], Fusion approach and Object-Z language are integrated in UML. In [Rag] a package to address real-time constraints is added to UML, but ambiguities in UML are not addressed.

2.1.4 Model Checking

Several researchers adopted approaches based on translation, in order to obtain a specification that can be used as the input of a model checker. An example is the translation 0 from statecharts [Har87] to Promela that allows to verify the properties of statecharts by means of the model checker SPIN [Hol97].

Silva Bastos and D'Almeida Sanchez proposed the translation of UML models into SDL (Specification and Description Language) [Bas01]. This kind of translation is however performed in a semi-automatic way because "UML models are informal and miss definitions that are normally required by SDL models".

vUML [Lil99] is a tool that translates UML models into Promela, so that the behavioural

properties of the models can be verified by means of SPIN. Note that Lilius and Porres Paltor provide their own definition of UML state machine (keeping the RTC semantics). A big limitation of vUML is that it is not able to verify linear temporal logic formulas, i.e., it does not deal with time, which is not a native concept in standard UML.

A translation of a suitable extension of UML into an axiomatic formal description is proposed by Alagar et al. [Ala99]. A tool developed by the same authors can be used to simulate the system, and the PVS theorem prover can be used to prove properties of the system. Of course in this case it is required that the user has the (non trivial) ability to use PVS.

Finally, an effective approach to real-time system development and verification is provided by the Taxys tool [Clo01]. Taxys works with Esterel [Ber92] programs annotated with temporal constraints, and contains the Kronos model checker for model analysis. The Esterel programs can be obtained from a variant of UML which features synchronous semantics by replacing StateCharts with SyncCharts. These charts are supported by tools (Esterel Studio).

2.1.5 Modelling Techniques for Product Families

A main concern is to find out a way to formally specify a product family based on the user requirements and to formally specify changes to user requirements. This will help to assess changes required on product family artefacts, and the impact of these changes, to meet user requirements. In this sense, there are several resources on current technology that can be used:

- **Product family:** system family techniques and methods defines some assets that can be useful for this topic, the domain model and decision model. The domain model defines the commonality and the variability for the product family. It defines all the possible final products. The decision model describes the open decisions for the product family. It contains the questions that need to be answered to obtain a concrete product from the product family.
- **XML:** XML technology can be used as a way to store customer requirements and flexible components. XML technology can be used to process customer requirements using the flexible components to obtain an instance of the system family. XML technology can be also used to constraint the customer requirements, so it can represent the decision model of a product family. This is, the set open decision and the set of values for these decisions that can be selected by the customer.
- **UML:** UML is a representation standard that is used to graphically define the structure, the functionality and the way in which this functionality is provided for a given system. The UML standard can be used to show both the product family and the concrete products.

2.2 State-of-the-Practice

2.2.1 Modelling Techniques and Formal Verification

State-of-the-practice in modelling: UML RT is used (typically via tools like Rose Real Time).

State-of-the-practice in development: little or no components are used. Practically no middleware (CORBA Real-time is still considered immature, and in any case it still little known). RTOS (like Vx Works) are used. The gap between models and the implementation is generally filled "by hand", inventing a mapping from the design elements to the actual

resources and interfaces provided by the RTOS.

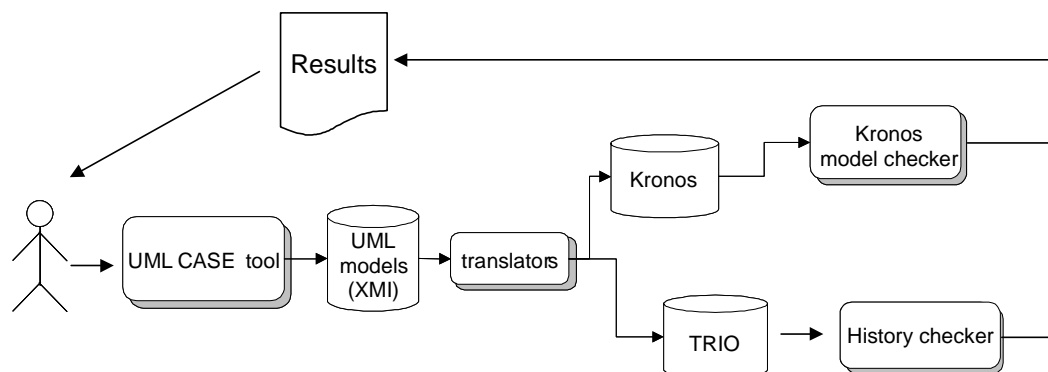


Figure 1. DESS specification construction and verification environment.

Concerning the specification and analysis of real-time systems, an example of state-of-practice is given by the results of the DESS project, which are illustrated in Figure 1. The construction of specifications is supported by a suitable notation (UML+, an extension of UML) [Lav01][Vie02] and by a graphic modelling tool which is an extension of Argo/UML [Arg02].

The UML+ specifications can be translated into TRIO [Ghe90] or into timed automata. This allows the usage of the TRIO tools (e.g., the history checker) or of the Kronos model checking tool [Yov97] (see Figure 2). The verification of the specifications is done in an automatic way, almost completely transparent from the user, which does not need to know the theory which lays behind TRIO or Kronos.

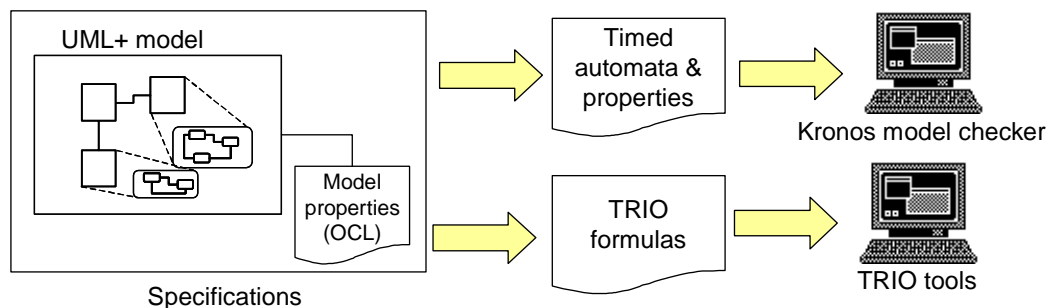


Figure 2. Usage of formal methods in DESS.

2.2.2 Modelling Techniques for Product Families

There are several approaches to formalise the product family artefacts. For example, the decision model can be specified using tables or FODA (Feature Oriented Domain Analysis) [SEI 1990] trees, the product design can be specified using UML or any other modelling notation. But there is no formalism in use for specifying the changes required, and the impact of these changes, when it is needed to evolve a product family.

2.3 Public Projects

A public project which is relevant for the topic is the DESS project [Des01], which is over. See Section 2.1.5 for a brief description of some relevant results of DESS.

ESI is currently working on CAFÉ project (<http://www.esi.es/Cafe/>). CAFÉ is devoted to the investigation of the product family theory. This project will provide deep knowledge on product families theory and the ways to apply it.

ESI is currently working on MASTER project (<http://www.esi.es/Master/>). MASTER is devoted to the instrumentation, enhancement, and refinement of the MDA approach. The results from this project will be useful to evaluate the way in which MDA supports UML models mapping.

2.4 Industry Needs

At the International Workshop on Industrial-strength Formal Techniques 1998 the panel on integrating formal and informal specification techniques [Bru98] provided a quite lucid picture of the situation concerning the practice of software specification in industry:

There is a strong demand from industry for rigor and formality. Academic research and tools vendors are asked to provide useful, powerful, cost-effective methods. Application of mathematically-based formal techniques requires much effort and skill on the part of the developer, and not many are willing to invest to acquire such skills. A technique that allows

developers to reap the benefits of rigorous analysis while using notations they are familiar with is more likely to be used in practice.

Although the statement reported above is three years old, it is clear that the situation has not evolved significantly: the offer of environment providing easy access to formal methods is still almost null, and the industrial practice is generally not relying on formal methods.

2.5 Research Questions

The ongoing research effort deals (as is clear from the description given in section 2.1) essentially with formal methods. However, typically they do not take into consideration:

- usability issues (i.e., the difficulty of applying formal methods);
- component-based development;
- evolution.

On the contrary, the commercial environments tend to incorporate CBD facilities, but are typically not formally defined and therefore do not support formal analysis or any other formal methods. Evolution is equally not taken into consideration.

The most relevant open issue (and the most relevant occasion for improving the development practice) is to put together the techniques that have been developed separately:

- UML based modelling notations;
- CBD facilities;
- formal methods.

Moreover, facilities for the management of evolution have to be considered.

2.5.1 Main Research Directions

A possible way for achieving the composition of methods mentioned above, while exploiting the outcome of the DESS project, and taking into due consideration components and

evolution is described in Figure 3.

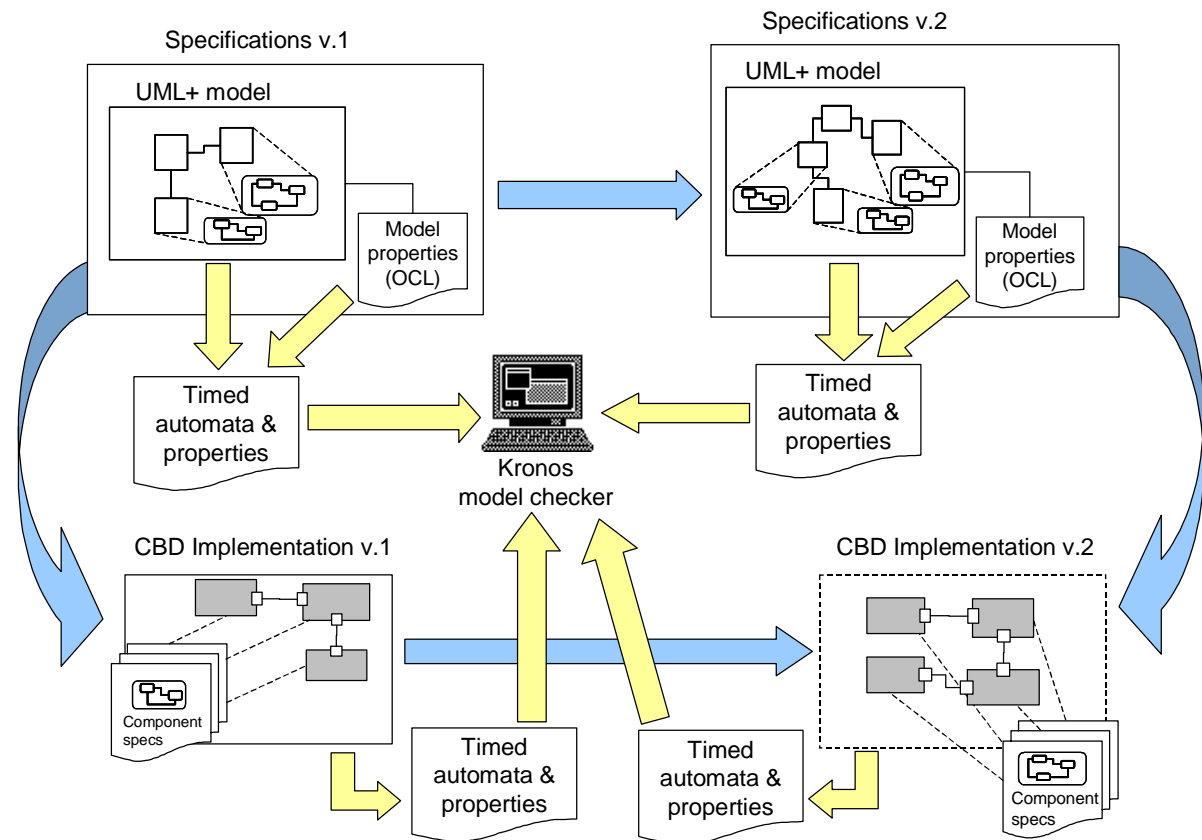


Figure 3. Global schematic view of the EMPRESS formal techniques.

The elements of the environments are:

- A tool (derived from the Argo/UML+ tool) will support modelling of new versions of specifications, with appropriate management of differences and reusing elements of previous specifications and/or library specifications of available components.
- A technique for deriving initial designs from specifications will be defined, together with supporting tools. The main problem here is the transition from the synchronous semantics of checkable models to the asynchronous semantics of the implementation (and of the real world).
- A techniques for deriving checkable specifications from component-based implementations (provided that the specifications of components and of the architectural support are available) will be defined. Supporting tools will be developed.

The environment resulting from the integration of these techniques is illustrated in Figure 4.

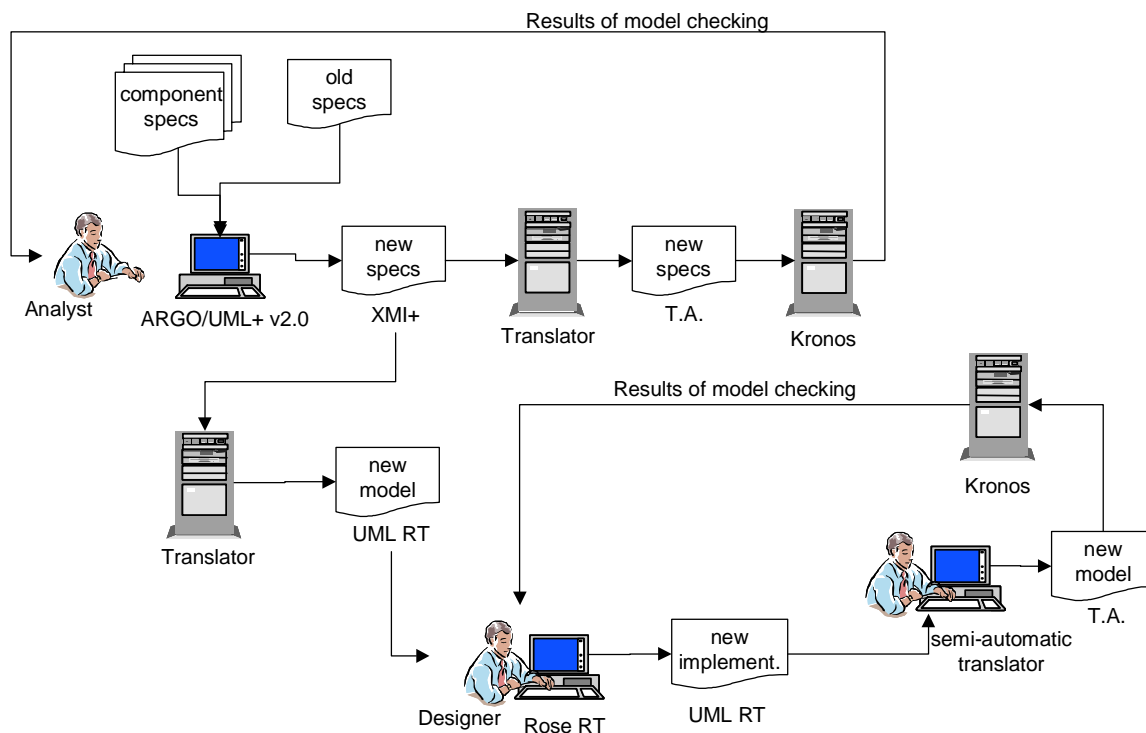


Figure 4. Global schematic view of the EMPRESS development and evolution environment.

2.5.2 Behavioural Subtyping

While the techniques dealing with model checking can provide a valuable contribution to the evaluation of properties concerning the synchronization and timing of the behaviour of the modelled system, they do not generally deal with the functional properties of the system. The reason is that model checking methods need to simplify the model they consider, in order to guarantee the decidability.

It is therefore interesting to complement the contribution of model checking with other methods that are more oriented to reasoning on the functional properties of the systems. Behavioural subtyping deals with using typing information to guarantee the behavioural compatibility of components that are subtypes (subclasses) of the “ideal” components which provide the specified behaviour.

Behavioral subtyping [Mey88] guarantees that all objects of a subtype preserve all of the original type’s invariants. In other words, any object of a subtype must be substitutable for an object of the original type without any effect on the program’s observable behavior. For pre- and post-conditions, the behavioral subtyping condition states that the pre-condition contracts for a type imply the pre-condition contracts for each subtype and the post-condition contracts for each subtype imply the post-condition contracts for the type.

The ability to reason about behavioural subtyping can be very beneficial in EMPRESS. In fact, we complement model checking methods with other techniques that allow us to deal with functional properties.

The goal of the research activity is to define techniques (based on either static analysis or run-time analysis) to support the verification of behavioural compatibility. The basis for these techniques is given by “programming by contracts”, which is particularly suited for component

based development.

2.5.3 Modelling Techniques for Product Families

2.5.3.1 Domain Specific Language

The goal is to identify and/or define a standard way to describe a domain. This language should include formalisms to describe all the components of the domain model: domain description, commonality and the variability within the domain, decision model, application model, and also the change process. In this sense, is there a product family description language? Are there description language for the several kinds of product family artefacts? Is possible to formalise the user requirements and their changes?

2.5.3.2 UML

Another goal is to find out the best way to represent product family artefacts before and after changes. In this sense, is UML capable of representing the changes between to models?

2.5.3.3 XML

XML can be used to standardise the way in which the information is shared between different systems. XML formalises the structure and the type of values that must be used to allow the communication between different parties. In the same way, can XML be used to formalise the product family artefacts (decision model and application model)?

3 Topic 2: Reasoning about the Change Process

This chapter analyses the change process topic from three points of view. First, it describes the state of the art around the technologies that can be applicable for the topic. In this part two main issues are addressed the modelling, notation and representation of the change process, and the configuration management process. Second projects that can be used as a source of information are listed. Finally, the open research questions and industry needs are addressed.

3.1 State-of-the-Art

3.1.1 Change Process Modeling – Notation – Representation

3.1.1.1 Derivation Process for the Change Management

Evolving a product family involves carrying out a set of tasks, using certain work products (e.g. change specification, product family artefacts, etc), producing a certain work products (e.g. adapted product family artefacts) and using certain resources, both infrastructure (e.g. configuration management tool) and people. All these aspects are defined in the evolution process for product families.

A main concern is to research how to formally describe and to implement the process to evolve a product family. In this sense, there are several resources that can be used:

Derivation process: there is some research already done in how to derive a final product based on a changing set of user requirements. Anyway, there is no notation to represent this process in a formal, clear and easy way to understand for a given product family.

SPEM: This metamodel is used to describe a concrete software development process or a family of related software development processes. SPEM is a UML profile that defines a notation to represent a process. This notation can be used to represent the change process.

3.1.1.2 Process Modelling Language

Research on software process and workflow management systems has produced several languages and systems particularly suited for process (or workflow) description and execution. Process Modelling Languages (PMLs) are similar in nature to programming languages [Ost87] and are usually executed by process engines, i.e., software systems which, given a process description, are able to partly automate and support the execution of the corresponding process.

In this context, several process modelling paradigms have been proposed and used, including Petri nets, grammars, rule-based systems, etc. These languages and the related supporting systems (usually capable of enacting the process) did not receive much attention from industry. The main causes were the difficulty of describing process models using these languages and the lack of support for distributed processes. Recent proposals have solved part of the problems of early process-centred environments, like the support of distributed development processes, the integration with tools and the coordination of process agents. Nevertheless, they are hardly used in industry.

While a relevant number of PMLs have been defined, none of them has been completely successful in addressing at the same time the requirements of both high-level modelling and enactment. In SLANG [Ban93] we have tried to address such requirements by identifying two levels of process representation: activity decomposition, concurrency, sequencing and synchronization among activities are modelled graphically, by means of high-level Petri Nets, while many other details such as association between activities, roles, and artefacts, mechanisms for interacting with the user, etc. are hidden in the implementation of the objects that are associated with tokens in the Petri net, and of the operations associated with transitions. Similar two-levels representations are also offered by several other PMLs, like – among others– Serendipity II [Gru98] and Juliette [Cas00].

Recently, UML has been also used as a PML [Eri00], [Jac99], [Jag99], [Mar00]. In fact, UML has some attractive features as a PML: it is popular, standard, graphical, equipped with several diagrams which support several views of the systems, extensible, supported by tools, provides a standard textual output (XMI), ... For instance, its authors used it to represent the Rational Unified Process (RUP) [Jac99], which is recommended for UML-based developments. However, UML has been conceived as a non-executable, semi-formal language. In other words, UML process models are suitable descriptions for humans, while they are neither sufficiently precise nor detailed to be interpreted by machines.

OPSS [Cug01] is a process support systems that originally employed Java as a PML. However, Java proved to be too low-level for several application contexts. In order to make process modelling and enactment available to all the numerous process managers who do not have the technical competence or the time to deal with details at the level of a programming language notation, UML was adopted as a language to describe processes not just for usage by humans, but also for process enactment [Nit02]. In this way the process modelers are allowed to write process descriptions by means of UML, and then convert these models into enactable OPSS models. By doing so UML is given a precise operational semantics and we make it enactable, while preserving its expressiveness as a high-level, human-oriented PML. The automatic translation of UML into the OPSS Java classes makes the approach quite efficient.

The description above applies to:

- Enactment-level descriptions. One can model a process in languages like Idef0 [IDE93], with the purpose of describing or documenting the process, but surely not at the detail level needed to automatically support the process enactment.
- General purpose process. Process languages defined so far are not specific for evolution. A relevant research area seeking to model the global software process as a feedback system and analyse its behaviour is being carried out by the group led by Prof. Manny Lehman [Leh85] [Leh87]. Their goal is to develop theories of software process and software evolution, producing models of such processes and the evolution phenomenon, applying the models to support software process improvement, providing tools for software release planning, management and so on.

3.2 State of the Practice

3.2.1 Change Process Modelling – Notation – Representation

3.2.1.1 Change Process Description

In the context of its software improvement activities, Philips Hasselt has created process and procedure descriptions for the following change processes:

- requirements engineering
- requirements management
- configuration management
- change management

The available change process descriptions are based on a generic template for process and procedure descriptions. The generic templates for process and procedure descriptions are based on the IDEF method for describing processes [IDE96].

However, the current change processes in Philips Hasselt have been developed with the software development activities for first-of-a-kind products in mind and they concentrate less on product family development.

3.2.1.2 Derivation Process for the Change Management

Currently the definition of any process is usually done based on an ad-doc notation. This makes more difficult the sharing and the understanding of the process by people not familiar with this notation.

Focusing in product families and processes that describe how to evolve them, there is no formal definition of the way in which a product family must be evolved.

3.2.2 Process Description Configuration Management

3.2.2.1 Configuration Management Process

As described for the sub-topic “Change Process Modelling – Notation – Representation”, Philips Hasselt has developed process and procedure descriptions for software configuration management. The configuration management and change management process and procedure descriptions have proven to be effective for the software development in several

set-top box-, DVD- and CD-RW projects.

Also as mentioned before, the current configuration management process concentrates on first-of-a-kind products and less on product family development.

3.3 Public Projects

ESI is currently working on CAFÉ project (<http://www.esi.es/Cafe/>). CAFÉ is devoted to the investigation of the product family theory. This project will provide deep knowledge on product families theory and the ways to apply it.

ESI is currently working on MASTER project(<http://www.esi.es/Master/>). MASTER is devoted to the instrumentation, enhancement, and refinement of the MDA approach. The results from this project will be useful to evaluate the way in which MDA supports UML models mapping.

3.4 Industry Needs

3.4.1 Versioning Management and Change Management

As stated in section 3.1.1.1, process modelling notations and enactment environments are available and are mature enough to support generic development process. The open issues concern the support of the evolution process.

In particular, interesting directions for improving the support provided by process environments to the evolution are:

- Integrating Software Configuration Management (SCM) tools in the process environment. The process manages software artefacts. when dealing with evolution it is essential that artefact versions are controlled. This is typically done by means of suitable SCM tools. It is therefore necessary that a process environment dealing with software evolution interacts in a smooth way with such tools.
- Reasoning on the process. Beyond process representation and enactment, it is increasingly important to be able to reason about the process in order to evaluate the cost of execution of the process, its efficiency, etc. Both static analysis (e.g., property assessment) and dynamic evaluation (e.g., simulation of the execution of the process) are to be investigated.
- Evolution-specific models. In order to make possible the process analysis, to support the enactment, or even just to give guidelines to the people involved in process evolution, evolution-specific process models are needed.

3.4.2 Product Family Evolution Process

The goal is to define a process to obtain different products based on a set of changing user requirements making use of the infrastructure provided by the product family. In this sense, which are the tasks to implement this process? Which are the inputs? Which are the outputs? How does the process address changes in the customer requirements?

3.5 Research Questions

3.5.1 Change Processes

The available process and procedure descriptions of the change processes in Philips Hasselt concentrate on the software development activities for first-of-a-kind-products. However, today more and more projects aim at developing products as part of a product family and especially aim at reuse and component based development.

The consequences of product family development for the change processes need to be investigated.

4 Topic 3: Techniques and Tools to Support the Change Process

4.1 State-of-the-Art

4.1.1 Change Estimation

When a new customer requests a software system belonging a product family, the owner of the product family needs to identify the changes necessary to meet customer needs and requirements to precisely estimate cost, time effort to produce the most competitive offer. In general, reasoning about management parameters when evolving a product family is a need for product family owners.

A main concern is to research on effective techniques to reason about management parameters when evolving a product family. In this sense, there are several resources on current technology that can be used:

Cost Models: currently there are many cost models that are able to estimate the effort and cost of a software development project based on some initial parameters. Examples of these cost models are: COCOMO, COCOMOII, Function points, Putnam,... These cost models are not applicable in product families; anyway they contain the cost drivers to take into account in any estimation.

XML: XML technology can be used as a way to store management information. Latter XML technology can be used to process these information to obtain reliable estimations.

4.2 State-of-the-Practice

4.2.1 Tool Support for Product Family Change Estimation

Currently there are some tools that support estimation for software development project. Anyway these tools are focused in traditional development, not in product family based development and evolution.

Focusing in product family there is no practice in the application of techniques to reason about the product family evolution. Estimations are currently made based on the Know-how and experience of people.

4.2.2 Current Process and Methods

Current product requirements within Philips CSI are gathered before the production phase. The requirements are being collected within one big document called the Functional Requirement Specification.

The (software) functional requirement specification is based on the commercial requirement specification, input from domain experts and specifications of former products.

Highest level of document is the commercial requirements specification written by the product manager, representing all clients and with good knowledge of the actual market situation.

Usually the functional requirements document is being produced by the software developers also responsible for the implementation

The functional requirements specification document is being signed off by all internal stakeholders at the start of the production phase.

All changes and additions to the original requirements documents have to be signed off by the members of a change control board.

Accepted changes require re-editing of the last version of the requirement document resulting in a younger version of the requirement document.

Experiences:

- Requirement documents were not brought under configuration management control.
- No tight relation between development / product versions and requirements
- The change control board is a good institute to prevent requirements creep during the production phase. All changes are being judged on necessity, development throughput time and costs.
- It is sometimes difficult to judge all consequences of proposed changes.
- Due to pressure from the product planning, developers put first priority on implementation resulting in requirements documents not reflecting the actual status.
- For each new product in the same product family a complete new requirements document is being written.

Tools:

No other tools as a text processor (Microsoft Office: Word) are being used nowadays.

Templates have been developed for the commercial and functional requirements documents and for the change proposals handled by the change control board.

4.2.3 Tools Supporting Change Management Process

The projects in Philips Hasselt use the the following tools for supporting their change management processes:

- Configuration management: Rational ClearCase
- The handling of change requests: Rational ClearQuest.
- Requirements management: limited experiments have been done with Rational RequisitePro.

However, in general the use of these tools concentrates on individual products. The use of these in product families development needs to be investigated.

For the estimation of management parameters as effort, cost and time no standard tools are in use. Dependent on the project, individual techniques or tools are used. Most of the estimations are based on history information and the top level design structure.

4.3 Public Projects

There are not public projects referring to this topic.

4.4 Industry Needs

4.4.1 Improvements to Current Processes

Needed improvements to current processes, methods and tools:

- Keep requirements actual. Add accepted changes to the requirements database and generate a new version of the requirements.
- Keep requirements versions in phase with the developed software versions. Know which requirements are applicable to a (software) product version.
- Changing requirements influence the design model, the implementation model and the test requirements. Define the relation between versions of those four process deliverables.
- Ensure traceability of requirements and the derived designs, implementations and test documents to see the consequences of changing requirements.
- Add associations to ensure traceability between requirements in same document and in other documents like e.g. commercial requirements specifications, use case documents, design (implementation) documents and test documents.
- Add unique identifiers for the individual requirements as associations require them.
- Provide project management feedback to monitor the project e.g. information which requirements are implemented and tested.
- Due to the big quantity of requirements and their relations, tools are needed.
- Those tools should not only be applicable to the requirement process deliverables but cover all development process deliverables including the design, implementation and test deliverables.

4.4.2 Configuration and Change Management Tools

The projects in Philips Hasselt need

As mentioned in 4.2, Philips Hasselt uses tools for configuration management and change request management, but these experiments were restricted to single product development projects. We need to investigate the use of these tools for product family development.

Requirements management tool support is considered to be very important in Philips Hasselt, but at this moment we are not ready to use e.g. RequisitePro in projects. The feasibility of using e.g. RequisitePro in our product family projects need to be investigated.

Also for cost estimation, currently based on individual experience and history information we need to investigate which tool and method support is possible and usable in our projects.

An important factor is the level of complexity of all the tools. On one hand, the processes and procedures are needed to improve the quality of work and deliverables, on the other hand, to be competitive in our solutions, one has to be low on overhead.

4.5 Research Questions

4.5.1 Product Family Change Management Information

The goal is to investigate the extra information to include in the product family to be able to assess the changes required and the impact of these changes to meet user requirements. This also includes the identification of appropriate ways to store, to process and to represent this information in order to make it useful for developers.

5 Conclusion

Literature

- [2UC02] 2U Consortium Unambiguous UML, <http://www.2uworks.org/documents.html>.
- [Ala99] Alagar, V.S., Muthiayen, D., Pompeo, F.: From behavioural specification to axiomatic description of real-time reactive systems. Fifth IEEE Real-Time Technology and Applications Symposium, Work-In-Progress Session. Vancouver, British Columbia, June 1999. IEEE Computer Society
- [Ame91] America, P. Designing an object-oriented programming language with behavioural subtyping. In Foundations of Object-Oriented Languages, REX School/Workshop, Noordwijkerhout, The Netherlands, May/June 1990, Lecture Notes in Computer Science, pages 60–90. Springer-Verlag, 1991.
- [Arg02] ARGO/UML, <http://argouml.tigris.org>
- [Ban93] Bandinelli S., Fuggetta A., and Ghezzi C., Software Process Model Evolution in the SPADE Environment, IEEE TSE Special Issue on Process Evolution, 19, 12, 1993.
- [Bas01] Silva Bastos, S.J., D'Almeida Sanchez, M.L.: Modelling Real Time Systems from Object-Oriented Models. IEEE Real-Time Embedded System Workshop, 2001
- [Ber92] Berry, G., Gonthier, G.: The Esterel Synchronous Programming Language: Design, Semantics, Implementation. Science of Computer Programming, Vol. 19, N. 2. (1992) 87—152
- [Bru] Bruel, J.M., and France, R., Transforming UML Models to Formal Specifications.
- [Bru98] Bruel, J.M., Cheng, B., Easterbrook, S., France, R. B., Rumpe, B.: Integrating Formal and Informal Specification Techniques. Why? How? Panel session, 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques 20-23 October 1998
- [Cas00] Cass A., Lerner B., McCall E., Osterweil L., Sutton S. Jr., and Wise A., Little-JIL/Juliette: A process Definition Language and Interpreter, in Proc. 22nd ICSE,

Limerick, Ireland, June 2000.

- [Cla01] Clark, T., Evans, A., and Kent, S., 2001, Engineering modelling languages: A precise meta-modelling approach, <http://www.2uworks.org/documents.html>.
- [Clo01] Closse, E., Poize, M., Pulou, J., Sifakis, J., Venier, P., Weil, D., Yovine, S.: TAXYS: a Tool for the Development and Verification of Real-Time Embedded Systems. Computer Aided Verification, CAV'01, Paris, France, July 23, 2001. LNCS 2102, Springer-Verlag
- [Cug01] Cugola G., Di Nitto E., and Fuggetta A., The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS, IEEE Transactions on Software Engineering, September 2001.
- [Des01] DESS project official Web site, <http://www.dess-itea.org>
- [Dou98] Douglass, B.P., 1998, Real-Time UML, Addison Wesley.
- [Eri00] Eriksson H.E., and Penker M., Business Modeling with UML, Wiley Computing Publishing, 2000.
- [Fea98] Feather, M.S., Dunphy, J., Rouquette, N.: Position paper for WITF 98
- [Fin01] Robert B. Findler, Mario Latendresse, Matthias Felleisen, "Behavioral Contracts and Behavioural Subtyping" Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE), Vienna, 10-14, September 2001.
- [Ghe90] Ghezzi, C., Mandrioli, D., and Morzenti, A., May 1990, TRIO, a logic language for executable specifications of real-time systems. The Journal of Systems and Software, 12, 2.
- [Gra00] Graw, G., Herrmann, P., and Krumm, H., 2000, Verification of UML-based real-time system designs by means of cTLA, In Object-Oriented Real-Time Distributed Computing (ISORC 2000) Proceedings, Third IEEE International Symposium on , 2000 Page(s): 86 –95.
- [Gru98] Grundy J., Apperley M., Hosking J., and Mugridge W., A decentralized Architecture for Software Process Modeling and Enactment, IEEE Internet Computing, 2, 5, 1998.
- [Har87] Harel, D.: Statecharts: A Visual Formulation for Complex Systems. Science of Computer Programming Vol. 8, N. 3. (1987) 231--274
- [Her97] Herrmann, P., Specification of Hybrid Systems in cTLA+, 1997, <http://citeseer.nj.nec.com/herrmann97specification.html>.
- [Hol97] Holzmann, G.J.: The Spin Model Checker. IEEE Trans. on Software Engineering, Vol. 23, N. 5, May 1997, 279-295
- [IDE93] Integration Definition for Function Modeling, IDEF0. Federal Information Processing Standards Publications, December 1993.
- [IDE96] IDEF method of describing the processes, ISBN 0471-13281-0, Process Mapping; how to reengineer your business processes, Daniel Hunt, 1996, chapter 6,p 96 –118, developed by Coopers & Lybrand
- [Jac99] Jacobson I., Booch G., and Rumbaugh J., The Unified Software Development Process, Addison-Wesley Object Technology Series, 1999.
- [Jag99] Jager D., Schleicher A., and Westfechtel B., Using UML for Software Process

- Modeling, In Proc. of ESEC/FSE'99, Toulouse, France, LNCS 1687, Springer, September 1999.
- [Kes92] Kesten, Y., and Pnueli, A., 1992, Timed and Hybrid Statecharts and their Textual Representation, Weizmann Institute of Science, In Formal Techniques in Real-Time and Fault-Tolerant Systems 2nd International Symposium.
- [Lam99] Lamport, L., 1992, Hybrid Systems in TLA+, <http://citeseer.nj.nec.com/lamport93hybrid.html>.
- [Lav01] Lavazza, L., Quaroni, G., and Venturilli, M., 2001, Combining UML and formal notations for modelling real-time systems, Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE), Wien, September 10-14.
- [Leh85] M M Lehman and L A Belady, Program Evolution - Processes of Software Change , Academic Press, New York, 1985, 552 pps. (ISBN 0-12-442441-4)
- [Leh87] Lehman M M, Process Models, Process Programs, Programming Support - Invited Response To A Keynote Address By Lee Osterweil, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 March 2 - Apr. 1987, IEEE Comp. Soc. pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 14 – 16
- [Lil99] Lilius, J., Porres Paltor, I.: vUML: a tool for verifying UML models. ASE'99. IEEE Computer Society (1999) 255—258
- [Lis94] Liskov, B. H. and J. M. Wing. A behavioural notion of subtyping. ACM Transactions on Programming Languages and Systems, November 1994.
- [Mar00] Marshall C., Enterprise Modeling with UML: Designing Successful Software through Business Analysis, Addison-Wesley, 2000.
- [Mey88] Meyer, B. Object-oriented Software Construction. Prentice Hall, 1988.
- [Mik98] Mikk, E., Lakhnech, Y., Siegel, M., Holzmann, G.J.: Implementing Statecharts in PROMELA/SPIN. IEEE Computer Society, October 21-23 1998
- [Nit02] E. Di Nitto, L. Lavazza, M. Schiavoni, E. Tracanella, M. Trombetta, Deriving executable process descriptions from UML”, ICSE 2002, International Conference on Software Engineering, Orlando, Florida, 19-25 May 2002.
- [OMG00] OMG, XML Metadata Interchange (XMI) Specification, Version 1.1, November 2000, <ftp://ftp.omg.org/pub/docs/formal/00-11-02.pdf>.
- [OMG01] OMG, Unified Modelling Language Specification, Version 1.4, September 2001.
- [Ost87] Osterweil L., Software Processes Are Software Too. Proc. 9th ICSE., ACM Press, New York, N.Y., 1987, pp. 2-13.
- [Pre02] The precise UML group, <http://www.cs.york.ac.uk/puml>.
- [Rag] Raghavan, G., and Larrondo-Petrie, M., A formal UML Package for Specifying Real-Time System Constraints.
- [SEI 1990] Kang, K, Cohen, S. “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, CMU/SEI-90-TR-021, 1990
- [Sel99] B. Selic, G. Gullekson, P.T. Ward, Real-Time Object-Oriented Modelling, Wiley, 1999.

- [Sou98] Francis D'Souza, D., and Cameron Wills, A., 1998, Objects, Components, and Frameworks With Uml: The Catalysis Approach, Addison-Wesley.
- [UML01a] Initial submission to ad/00-09-02 (UML 2.0 Superstructure), 22/10/2001, <http://www.2uworks.org/documents.html>
- [UML01b] Initial submission to ad/00-09-01 (UML 2.0 Infrastructure) ad/00-09-03 (UML 2.0 OCL), 22/10/2001, <http://www.2uworks.org/documents.html>
- [Vie02] Vieri Del Bianco, Luigi Lavazza, Marco Mauri, A Formalization of UML Statecharts for Real-Time Software Modelling", The Sixth Biennial World Conference on Integrated Design Process Technology (IDPT 2002), Pasadena, California, 23-28 June 2002.
- [Yov97] Yovine, S., October 1997, Kronos, A Verification Tool for Real-Time Systems (Kronos User's Manual Release 2.2), Springer International Journal of Software Tools for Technology Transfer, Vol. 1.