# IPTES: A Concurrent Engineering Approach for Real-Time Software Development

P. Pulli
Technical Research Centre of Finland (VTT)
Computer Technology Laboratory
Kaitovayla 1, P.O. Box 201, SF-90571 Oulu, Finland

R. Elmstrøm
The Institute of Applied Computer Science (IFAD)
Forskerparken 10, DK-5230 Odense M, Denmark

**Abstract**

The constantly increasing concurrency, complexity, and risks associated with the industrial development of real-time embedded computer systems has been approached in different ways in recent years. In Esprit project no. EP5570, called IPTES, a methodology and a supporting environment to support the Boehm's spiral process are being developed. The prototyping environment will enable the specification, development, and verification of executable system models so that different parts of the system may represent different modeling levels and yet can be executed as a total system. Concurrent engineering problems in connection with multi-supplier, distributed software development are also addressed in the IPTES environment. In the IPTES project the concept of heterogeneous prototyping is proposed as a solution. Each of the development teams may use relatively abstract models of the other parts of the systems as a testbed (environment model) for their own part, yet they can proceed developing their own part full speed by means of advancing the maturity of their part to the next abstraction level(s). The IPTES environment provides a set of tools to help in the process of creating, analysing, and testing distributed heterogeneous prototypes.

## 1 Introduction

Three major trends can be seen in the development of software for embedded systems in the nineties:

1. The complexity of systems is constantly increasing.
2. Just-on-time delivery policies and flexible manufacturing will require new degrees of freedom from the traditional software development process.
3. The number and the impact of risks associated with software development are on the increase.

In general, the software part of typical embedded systems is becoming more dominant. Therefore the management of risks, such as misunderstanding of customer needs, subcontractor coordination problems, schedule overruns, staff or budget overruns, functional or quality pitfalls, or in the worst case cancelled projects is increasingly important in the industrial development of embedded systems.

Over the last few years, more and more attention has been paid to alternative software development models that could both overcome deficiencies [Agresti86] of the traditional waterfall model [Boehm81], and accommodate activities such as prototyping, reuse, and automatic coding as part of the process.

In ESPRIT project EP5570, IPTES[1], a methodology and a supporting environment to support Boehm's spiral process are being developed. The IPTES methodology and the supporting environment integrates the use of Ward and Mellor's Structured Analysis for Real-time Systems, the formal specification language VDM-SL, and incremental prototyping into Boehm's spiral model. In this paper we present the IPTES methodology.

In the next section we present the background of the IPTES methodology in the spiral process model. In section 3 we present how the IPTES methodology supports the use of the spiral model in the development of embedded software systems. In section 4 we present the benefits foreseen in applying the IPTES approach. In section 5 we present some related work and finally in section 6 we give some concluding remarks.

## 2 Spiral Process Model

The spiral model proposed by Boehm [Boehm88] (Figure 1) is a development process model in which prototyping and reuse are important ingredients. The model describes an iterative development process where planning, risk identification and resolution and development (of prototypes or product) are part of each iteration.

The risk-driven nature of the spiral model makes it particularly applicable to complex embedded systems.

### 2.1 Risk Management

The spiral model guides a developer to postpone detailed elaboration of low-risk software elements and to avoid going too deep in their design until the essential high-risk elements of the design are stabilised. Risk management requires appropriate attention to early risk resolution techniques such as early prototyping and simulation. The spiral model may incorporate prototyping as a risk reduction option at any stage of development and explicitly calls for suitable risk assessment and risk control activities throughout major portions of the development process.

[1]IPTES is an acronym for *Incremental Prototyping Technology for Embedded real-time Systems*

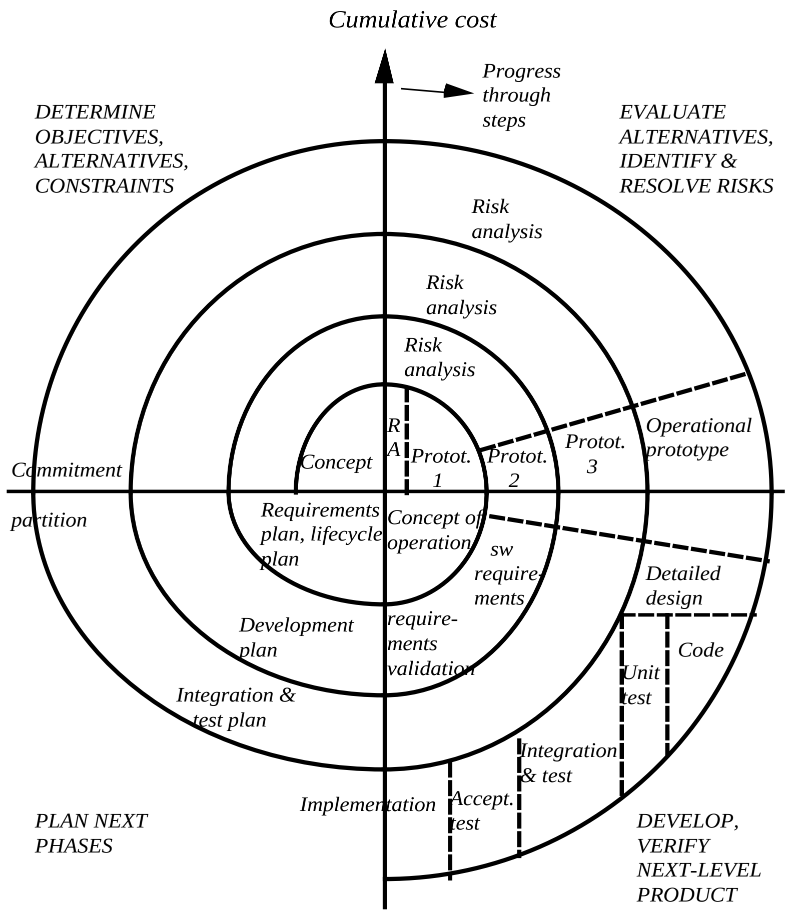


Figure 1: Boehm's spiral model for the software development process. The radial dimension represents the cumulative cost, and the angular dimension represents the progress made in completing each cycle of the spiral.

Risk management involves the following steps [Boehm91]:

- Risk assessment techniques
  - Risk identification produces lists of the project specific risk items likely to compromise a project's success.
  - Risk analysis quantifies the loss probability and loss magnitude for each identified risk item, and it assesses compound risks in risk item interactions.
  - Risk priorisation produces a ranked list of risk items according to their severity.
- Risk control techniques
  - Risk management planning helps prepare you to address each risk item. It also includes the coordination of the individual risk item plans with each other and with the overall project plan.
  - Risk resolution produces a situation in which the risk items are eliminated or otherwise resolved.
  - Risk monitoring involves tracking of the project's progress towards resolving its risk items and taking corrective action where appropriate.

## 2.2 Concurrent Threads of Activities

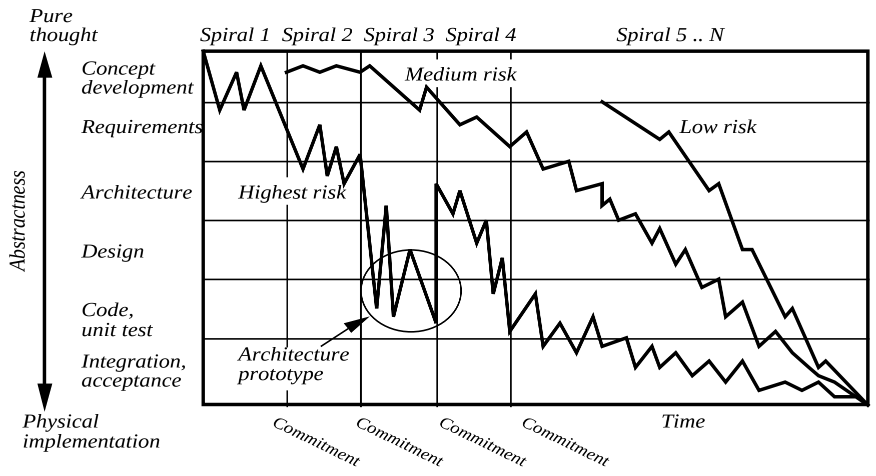


Figure 2: Level of abstraction vs. time under the spiral model.

The spiral model allows concurrent threads of development activities that may traverse the traditional progression of software product phases in a loosely synchronised manner. The concurrent threads may be organised around levels of risk [Boehm88]. Figure 2 gives an example [TRW89] of concurrent development threads in a hypotethical development project targeted to a major breakthrough product. In Figure 2, the horizontal axis represents elapsed time. The vertical axis represents the level of abstraction at which a representation or an understanding of the system is being developed. The development process is depicted by three traces through the two dimensional space. Each trace corresponds to a thread of engineering activities. In general, moving downwards represents progress. A highly jagged trace represents an activity thread in which much iteration (prototyping) takes place. This is seen in Figure 2 where an architectural prototype is developed for the high-risk thread, during the third spiral. This prototype is later analysed and the produced code thrown away. Based on the lessons learned from the architectural prototype, the high-risk thread enters full-scale design and implementation based on the valid parts of the architecture and design during Spiral 4. Concurrent with the third and fourth spirals the medium-risk elements are being specified. During the fifth and later spirals the high-, medium- and low-risk threads progress concurrently, leading to incremental integration, installation, and use.

## 3 IPTES Approach

The spiral model is a generic model, so it does not explicitly define the milestone abstraction levels to be produced for each cycle. The spiral has to be customised on a company or project basis. [Royce90] presents a derivative of the spiral model which explicitly defines these milestones according to the US military standard [DOD-STD-2167A]. [Nettles91] describes another on-going coordinated effort of producing guidelines for applying Spiral model which also heads for DOD-2167A compliance. We have chosen not to strive for 2167A compliance, because it being a document driven approach tends to force excess synchronisation of concurrent development threads.

| *MODEL* | *SUB-MODEL* | *IMPLEMENTATION DEPENDENCE* | *OBSERVABILITY OF BEHAVIOUR* | *NOTATION* | *DEVELOPENT PHASE* | *COMMENTS* |
|---|---|---|---|---|---|---|
| *Logical* | *Environmental* | *Independent* | *Non-transparent* | *Context diagram, events list* | *Analysis* | *System scope focusing on external events* |
| | *Behavioural* | | | *DFD, STD, ERD, MetaIV mini-specs* | | *Implementation-free user-observable requirements* |
| *Physical* | *Subsystems* | *Dependent* | | | | *System modularity, concurrent engineering* |
| | *Processor Environment* | | | | *Design* | *Processor configuration, interfaces* |
| | *Software Environment* | | *Transparent* | | | *Architecture, execution and storage units* |
| | *Code Organisation* | | | *Structure chart, OOSD* | | *Code, class/object architecture, and interfaces* |
| *Implementation* | | | | *Code* | *Implementation* | *Reuse, run-time adaption* |

Figure 3: The different levels of abstraction for real-time system development proposed by the IPTES guidelines.

Figure 3 presents the abstraction levels we propose. They follow the abstraction levels of a well-known embedded system development method, Ward & Mellor's Structured Analysis for Real-Time Systems (SA/RT) method [Ward&85] enhanced with a subsystems level of abstraction similar to one proposed in [Gomaa89]. This extension may prove useful for large systems development and for concurrent engineering [Reddy&92] purposes. The rationales behind choosing Ward&Mellor SA/RT like levels of abstraction are:

- SA/RT being a model-oriented rather than phase- or document-oriented method it supports several abstraction levels.

- SA/RT is a well-known and understood method used in industrial embedded system development, i.e. it is not too fancy and risky.
- There exists textbooks, courses and consulting services providing an abundance of guidelines, heuristics and other support.

The Logical Model (LM) consists of an environmental model and a behavioural model. In the environmental model the system's environment and the events from this environment are described. In the behavioural model the system's externally observable behaviour is described. The physical model consists of several sub-models. The optional Subsystems Model (SM) is used to identify subsystems within a logical model. Subsystems are characterised with high internal cohesion and low external coupling. Subsystem model is useful for large logical models, and for systems where parts of the model have multiple instances[2]. The Processor Environment Model (PEM) describes how system activities and data are allocated to different processors which are truly concurrent. The interfaces between processors are also described. The Software Environment Model (SEM) is a description of the software architecture inside one processor. The SEM model describes how concurrency will be solved using a sequential processor. The Code Organisation Model (COM) describes the modularisation scheme to implement the software. It will identify a hierarchy of modules and data structures. For object oriented designs it will identify a hierarchy of classes and objects. The Implementation Model (IM) describes the product implementation.

### 3.1 Heterogeneous Prototype

A heterogeneous prototype is an executable system model whose different parts may be specified at different abstraction (modeling) levels, and yet they can be executed together as a total system. Over the lifetime of the prototype the mix of abstraction levels may change [Gabriel89]. Figures 4 and 5 [Mortensen90] illustrate the concept of the heterogeneous prototype.

### 3.2 Incremental Prototype

Incremental prototyping is the process of building heterogeneous prototypes over time. The incremental prototyping process may contain concurrent engineering [Reddy&92], i.e. there may be several teams working simultaneously with different heterogeneous prototypes. Each of the teams may use relatively abstract models of the other parts of the system as a testbed (environment model) for their own part, yet they can proceed developing their own part full speed by means of advancing the maturity of their own part to the next abstraction level.

Concurrent engineering can take place at the level of concurrent threads shown in Figure 4, or it may take place at a subsystem level, i.e. work for each subsystem may contain concurrent threads, see Figure 6.

### 3.3 Decision Support Techniques

The spiral model decision making has following characteristics:

- Planning (at all levels) is carried out using the standard planning sequence: determine objectives, alternatives, constraints, evaluate alternatives, identify and resolve risks.
- Significant parts of the global decision making activities can explicitly be left to be taken later in the project.

[2]Subsystem can also be thought as a unit of development work to be allocated to working groups within a large project.

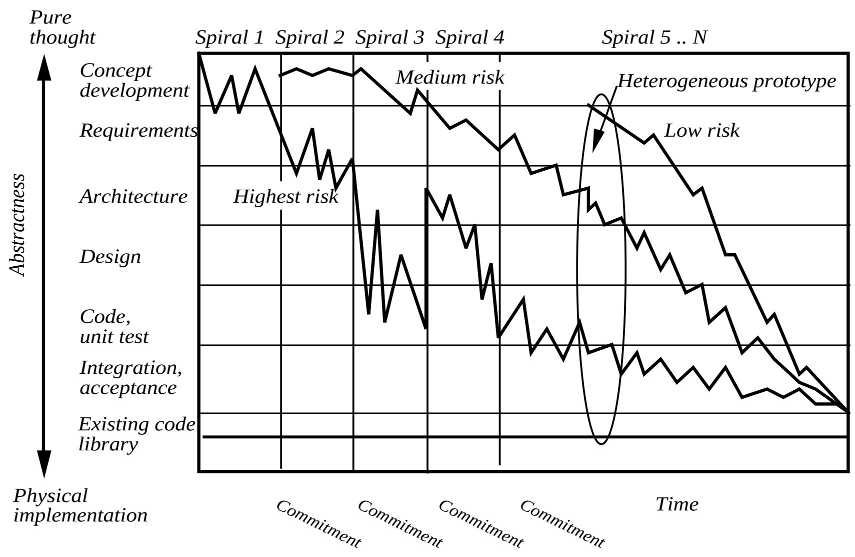


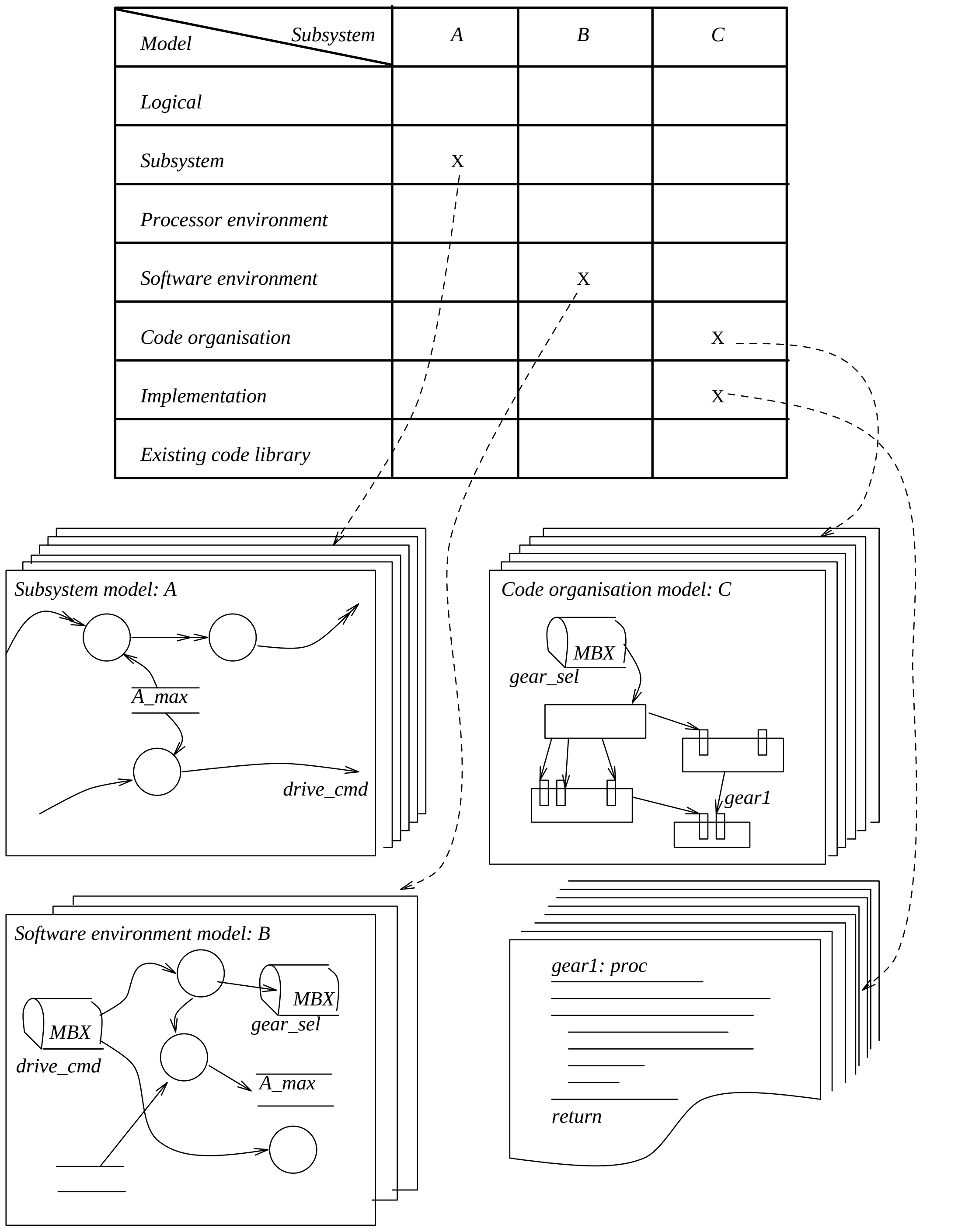
Figure 4: A heterogeneous prototype can be viewed as a vertical snapshot of concurrent development threads of Boehm's spiral model.

- There are explicit provisions for contingency planning activities during the project.
- Concurrent engineering aspects: There is relative freedom in the order/concurrency tasks are carried out.

In IPTES we have chosen to support decision making with an advanced form of value analysis especially suited for teamwork. Quality Function Deployment (QFD) [Zultner89] was developed in Japan in 1970's and 1980's [Akao90] as an integrated set of quality tools and techniques. It is used for market research and product design purposes to make explicit the "voice of the customer" throughout the product design process. Basic to the application of QFD is the use of variety of matrices to examine in detail the interaction of various dimensions such as function, cost, customer demands, raw materials, etc. Matrices are prepared by a team effort, so they have the potential to bridge the expertise of different individuals or groups. We see the QFD techniques as complementing the Boehm spiral model in teamwork-based decision support. From the decision making point of view, the QFD matrices present a roadmap for the actual decision network [Curtis87] of the project team. We are currently looking for ways to extend the current QFD system [King89] for identification and quantification of risk elements of the product, project, and the company infrastructure.

## 3.4 Specification Languages

We have chosen Ward and Mellor's Structured Analysis for Real-Time Systems (SA/RT) graphical notation as a basis for specification and design descriptions in IPTES. However, SA/RT being a semi-formal language implies that an executable dialect of the SA/RT notation has to be developed since the use of incremental prototyping requires the use of executable specification languages. In the IPTES project we have chosen to use VDM-SL [Elmstrøm&93a] and high-level timed Petri nets [Felder&93] to give exact formal syntax and semantics to SA/RT.



| Model \ Subsystem | A | B | C |
|---|---|---|---|
| Logical | | | |
| Subsystem | X | | |
| Processor environment | | | |
| Software environment | | X | |
| Code organisation | | | X |
| Implementation | | | X |
| Existing code library | | | |

Figure 5: An example of a heterogeneous prototype. Subsystem C is already partly coded while the other subsystems are modeled at higher levels of abstraction. Notice that models communicate through shared elements, such as data-flows, data-stores, operating system communication primitives, and procedure calls.

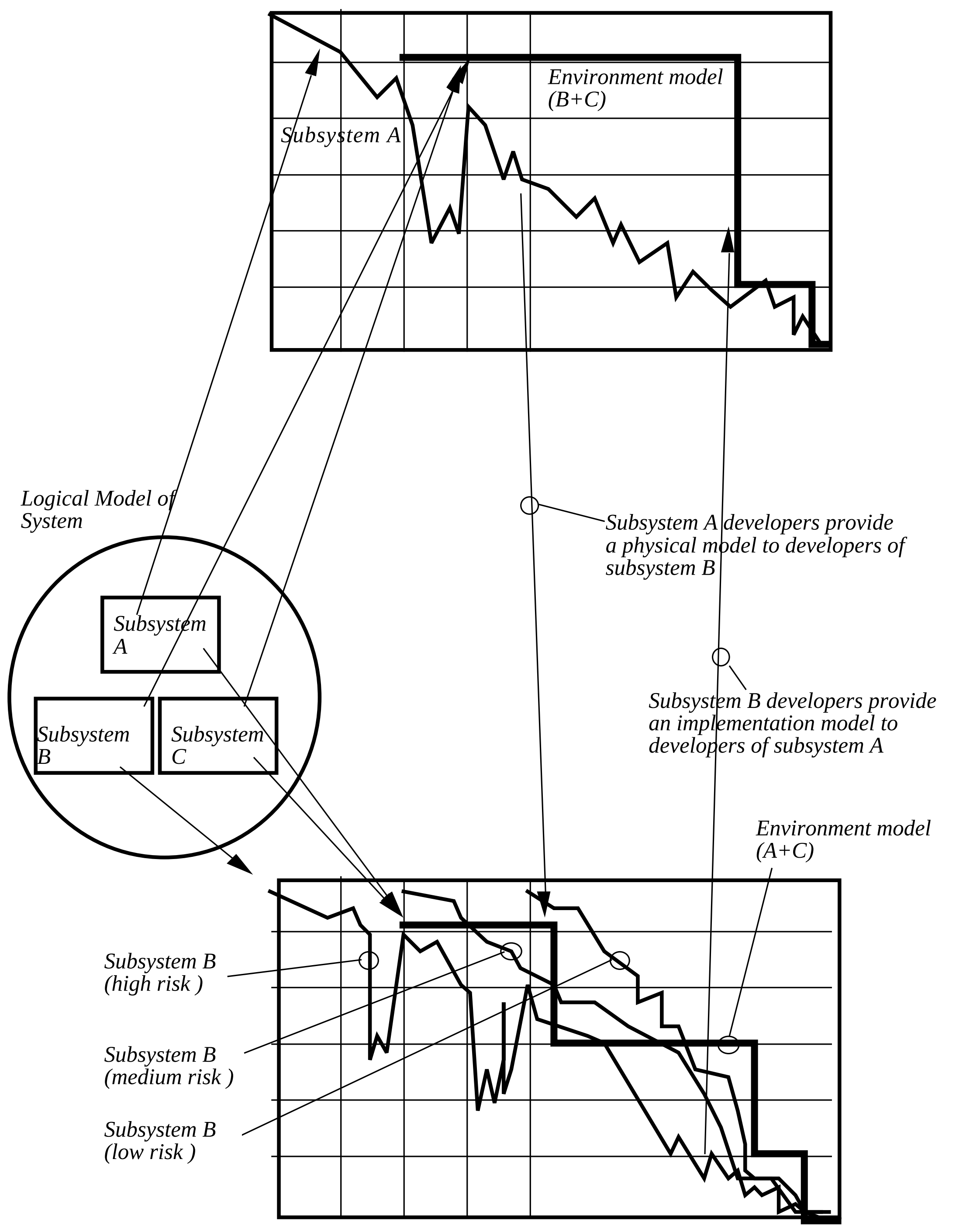

Environment model (B+C)
Subsystem A
Logical Model of System
Subsystem A
Subsystem B
Subsystem C
Subsystem A developers provide a physical model to developers of subsystem B
Subsystem B developers provide an implementation model to developers of subsystem A
Environment model (A+C)
Subsystem B (high risk )
Subsystem B (medium risk )
Subsystem B (low risk )

Figure 6: An example of incremental prototyping in a concurrent engineering context. Several loosely coupled heterogeneous prototypes are used during development to allow maximum concurrency of working groups.

We have also defined an executable subset of the emerging BSI and ISO standard VDM-SL language [Elmstrøm&93b] to be used for specifying the data transformation part of SA/RT in mini-specifications. The derived mini-specification language is called IPTES Meta-IV.

A high-level timed Petri net (HLTPN) kernel is used as the underlying execution mechanism of the IPTES environment. This kernel is used as basis for the execution and analysis of these SA/RT-VDM models. The HLTPN kernel makes the IPTES environment open for extension to other executable specification languages.

Run-time adaptation techniques [Leon&93] define a basic set of distributed data, control, and timing exchange mechanisms between the HLTPN kernel and high-level programming language routines.

### 3.5 Tool Environment

The IPTES environment [Leon&93] provides a set of graphically-oriented tools to help in the process of creating, analysing and testing distributed heterogeneous prototypes. The visualisation of prototype execution is based on graphical animation techniques [Pulli&93].

Internally, the environment is based on a representation of the system in terms of high-level timed Petri nets with shared places [Puente&93]. The interchange of information between the nodes is performed by a real-time object communication subsystem that ensures consistency between shared places.

## 4 Benefits

The following benefits are foreseen in applying the IPTES approach from the viewpoint of project managers and software engineers:

- Efficient user needs tracking and accommodation into product
- Improved project visibility
- Efficient coordination of subcontracting
- Management of increasing complexity
- Piecewise modernisation of a mature product
- Controlled transfer of technically and commercially risky features into products
- Improved control over time within the project
- Advanced quality management
- Harnessing application code generators in development work

Figure 7 describes how the IPTES mechanisms support these benefits.

## 5 Related Work

Gabriel presents the requirements for a future prototyping environment in [Gabriel89]. Gabriel foresees the need for heterogeneous prototyping by requiring that elements from behavioural and structural prototypes can be combined and during the development process this mix may change as the requirements to the prototype change.

| Benefit \ Mechanism | | Modeling | | | Prototyping | | | | | | Decision support | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Abstraction | Model libraries | Formal notations | Executable models of subsystems | Executable environment models | Executable models used for acceptance test | Run-time adaptation to existing C code | Heterogeneous prototypes | Distributed prototypes | Customer feedback on prototypes | QFD decision support | Risk and dependency analysis | Detection of critical parts |
| Communication | User needs accomodation | | | | △ | △ | | | | | ● | ● | | |
| | Improved visibility | | | | ○ | ● | | △ | | | ● | ○ | | |
| | Coordination of subcontracting | | | | △ | △ | ● | ● | ○ | | | | | ● |
| Activity structuring | Management of complexity | ● | ○ | ● | | | | | ● | ○ | | △ | | |
| | Piecewise product modernisation | | | | ● | | | ● | ○ | ● | | | | |
| | Transfer of risky features to product | | | | △ | | | | | | ○ | | ● | ● |
| Insfrastructure | Control over development time | | | | | | | | | | ○ | ○ | ● | ● |
| | Quality management | ○ | | ● | | | △ | | △ | | ● | ● | △ | △ |
| | Application code generators | | | ● | | | | ● | | | | | | |

Correlation: ● strong ○ medium △ weak <empty> none

Figure 7: Benefits supported by IPTES mechanisms. The rows of the matrix describe the individual benefits of the IPTES approach. The columns describe the mechanisms supporting the benefit.

Luqi has presented a prototyping environment for large software system design based on reusable Ada software components [Luqi86]. The computational model is based on data flow under semantically unified control and timing constraints. Luqi has presented the importance of the computational model for a prototyping tool, language and method in [Luqi86]. A limitation of the Luqi's system is that it does not support multiple abstraction levels. The abstraction level supported is roughly equivalent to the software environment model of SA/RT[3].

Harel et al. have produced a commercial, graphical executable specification tool, Statemate [Harel&90], that has prototyping features. It is possible to automatically generate prototype code from the activity-chart and Statechart [Harel87] specifications. Currently translations into Ada and C code are supported. A limitation of the Statemate tool is that it does not support multiple abstraction levels. Currently, an abstraction level equivalent to the logical model of SA/RT is supported[4]. However, it is possible to combine the prototype code generated out of Statemate models with user-written programs in Ada or C [Harel&90], [Coleman&90].

A recently started EUREKA project "RiskMan" [Manperil&91] is working on defining a project management methodology and developing a toolset for the management of large, predominantly software-oriented systems. The project is developing modules matching the spiral model iterative life cycle steps. The project has selected value analysis as a decision support technique to be applied for identification of objectives and alternatives, for traceability, and for various constraints.

Aoyama [Aoyama87] and Hatley [Hatley91] have studied the requirements for organisation infrastructure to support concurrent development of embedded software-intensive systems.

[Blumofe&88] and [Cadre90] describe a commercial, graphical executable specification tool, Teamwork/SIM. This tool supports a limited form of execution, capable of expressing only control and timing issues. Computations and data have been omitted. However, Teamwork/SIM supports several abstraction levels equivalent to the LM, PEM and SEM from SA/RT.

There are a number of tools for SA/RT logical model execution [Webb&86], [Reilly&87], [Coomber&90], [Athena89].

Some of the surveyed tools can be used for execution of heterogeneous models, and in that sense they make incremental prototyping possible. However, none of the tools supports incremental prototyping[5].

## 6 Concluding Remarks

The IPTES approach addresses the future requirements of the software engineering process: overcoming increasing complexity and development risks and supporting flexible just-on-time product development:

- IPTES improves communication by improving the visibility of the software engineering work, allowing user needs to be accommodated earlier, more reliably, and accurately.

[3]One can argue that there are in fact two abstraction levels: the Software Environment Model level and the code level. This is because Luqi's system does not have a mini-specification language; instead Ada is used.

[4][Harel&90] mentions plans to make the prototype code generation more adjustable. This can be interpreted as an indication of interest to support more physical abstraction levels.

[5]To better understand the difference between "makes possible" and "supports", consider the case of object-orientation. Any programming language can be used for constructing object-oriented software. However, the benefits of object-orientation were first realised with proper languages and environments like Smalltalk-80 [Goldberg&85].

- IPTES allows efficient development work structuring and allocation to concurrent development teams and individuals. IPTES allows teams to use intermediate results from other teams for validating their own progress.
- IPTES strengthens the organisational infrastructure by providing better control over development time, subcontracting, and product quality elements.

These benefits are achieved with strong modeling, prototyping and decision support capabilities.

## 7 Acknowlegdements

The work presented in this paper has been carried out in the IPTES project [Pulli&91b]. The objective of the IPTES (Incremental Prototyping Technology for Embedded Real-Time Systems) project is to develop methodologies, tools, and theoretical results on distributed prototyping of real-time systems. IPTES is partially funded by the European Communities under the ESPRIT programme, project no. EP5570 and the Technical Development Centre of Finland (TEKES).

The IPTES consortium is formed by IFAD (Denmark), VTT (Finland), MARI (UK), CEA/LETI (France), ENEA (Italy), Synergie (France), Universidad Politécnica de Madrid (Spain), Telefónica I+D (Spain), Politecnico di Milano (Italy), and Rautaruukki (Finland).

## 8 References

[Agresti86] W. Agresti (editor). New Paradigms for Software Development. IEEE Computer Society Press, 1986. 295 pages.

[Akao90] Y. Akao. Quality Function Deployment. Integrating Customer Requirements into Product Design. Productivity Press, Cambridge, Massachusetts Norwalk, Connecticut, 1990. 369 pages.

[Aoyama87] M. Aoyama. Concurrent Development of Software Systems – A New Development Paradigm. ACM Sigsoft Software Engineering Notes, 12(3):20–23, 1987.

[Athena89] Athena Systems Inc. Foresight: Modeling and Simulation Toolset for Real-Time System Development, User's Manual. March 1989.

[Blumofe&88] R. Blumofe, A. Hecht. Executing Real-Time Structured Analysis specifications. ACM Sigsoft Software Engineering Notes, pages 32–40, 13, 3 1989.

[Boehm81] B. Boehm. Software Engineering Economics. Prentice-Hall, 1981. 767 pages.

[Boehm88] B. Boehm. A Spiral Model of Software Development and Enhancement. IEEE Computer, 21(5):61–72, 1988.

[Boehm91] B. W. Boehm. Software Risk Management: Principles and Practices. IEEE Software, pages 32–41, January 1991.

[Cadre90] Cadre Technologies Inc. Teamwork/SIM. User's Guide. Release 4.0. December 1990. Part Number D048XX4A.

[Coleman&90] G.L. Coleman, C.P. Ellison, G.G. Gardner, D.L. Sandini, J.W. Brackett. Experience in Modelling a Concurrent Software System using Statemate. Proceedings of the Compeuro'90 Conference. Tel Aviv, Isreal, pages 104–108, Washington D.C. IEEE Computer Society Press, May 1990.

[Coomber&90] C. Coomber, R. Childs. A graphical tool for prototyping of real-time systems. ACM Sigsoft Software Engineering Notes, pages 70–82, 15, 2 1990.

[Curtis87] B. Curtis. Models of Iteration in Software Development. Proc. 3rd International Software Process Workshop, Breckenridge, Colorado, pages 53–56, 17-19 November 1987.

[DOD-STD-2167A] Military standard DOD-STD-2167A: Defence System Software Development. Department of Defence (US), February 1988. 51 pages.

[Elmstrøm&93a] R. Elmstrøm, R. Lintulampi, M. Pezze. Giving Semantics to SA/RT by Means of High-Level Timed Petri Nets. This issue of Real-Time Systems Journal, 1993.

[Elmstrøm&93b] R. Elmstrøm, P.B. Lassen, M. Andersen. An Executable Subset of VDM-SL, in an SA/RT Framework. This issue of Real-Time Systems Journal, 1993.

[Felder&93] M. Felder, C. Ghezzi, M. Pezze. High-Level Timed Petri Nets as a Kernel for Executable Specification. This issue of Real-Time Systems Journal, 1993.

[Gabriel89] R.P. Gabriel (editor). Draft Report on Requirements for a Common Prototyping System. ACM Sigplan Notices, 24(3):93–165, 1989.

[Goldberg&85] A. Goldberg, D. Robson. Smalltalk-80: The Language and its implementation. Addison Wesley, Reading Massachusetts, 1985.

[Gomaa89] H. Gomaa. A Software Design Method for Distributed Real-Time Applications. The Journal of Systems and Software, 9(2):81–94, 1989.

[Harel87] D. Harel. StateCharts: A Visual Formalism for Complex Systems. Science of Computer Programming, 8(3):231–274, 1987.

[Harel&90] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, M. Trakhtenbrot. STATEMATE: A Working Environment for the Development of Complex Reactive Systems. IEEE Transactions on Software Engineering, 16(4):403–414, April 1990.

[Hatley91] D.J. Hatley. Parallel, Cooperative System Development. Proceedings of Embedded Systems Conference, Santa Clara, California, Vol II, 823–836, September 24-27 1991.

[King89] B. King. Better Designs in Half the Time: Implementing QFD Quality Function Deployment in America. Goal/QPC, Methuen, Massachusetts, 1989. 564 pages.

[Leon&93] G. León, J.A. de la Puente, J.C. Dueñas, A. Alonso, and N. Zakhama. The IPTES Environment: Support for Incremental, Heterogeneous and Distributed Prototyping. This issue of Real-Time Systems Journal, 1993.

[Luqi86] Luqi. Rapid prototyping for large software system design. Ph.D thesis, University of Minnesota, 103 pages, 1986.

[Manperil&91] C. Manperil, Y. Rommel, J. Prinz. Risk Management Applied to Predominantly Software-Oriented Systems: The RiskMan Project. Draft paper, available from CGI/CR2A, 19 Avenue Dubonnet, 92411 Courbevoie Cedex.

[Mortensen90] B. G. Mortensen (Coordinating Proposer). IPTES: Incremental Prototyping Technology for Embedded Real-Time Systems. Part II. Project Description. Technical Report, IFAD, Odense, Denmark, January 8, 1990.

[Nettles91] D. Nettles. Consortium Prepares Evolutionary Spiral Process Deliverables. SPC Quarterly, pages 4–6, Spring 1991.

[Puente&93] J.A. de la Puente, A. Alonso, G. León, J.C. Dueñas. Distributed Execution of Specifications. This issue of Real-Time Systems Journal, 1993.

[Pulli&91b] P. Pulli, R. Elmstrøm, G. León, J.A. de la Puente. IPTES– Incremental Prototyping Technology for Embedded real-time Systems. ESPRIT Information Processing Systems and Software, Results and Progress of Selected Projects 1991, pages 497–512, Esprit, Commission of the European Communities, November 1991.

[Pulli&93] P. Pulli, M. Heikkinen, R. Lintulampi. Graphical Animation as a Form of Prototyping Real-Time Software Systems. This issue of Real-Time Systems Journal, 1993.

[Reddy&92] R. Reddy, R.T. Wood, K.J. Cleetus. The DARPA Initiative in Concurrent Engineering. Concurrent Engineering Research in Review, pages 2–10, Winter 1991/1992, 1992.

[Reilly&87] E.L. Reilly, J.W. Brackett. An experimental system for executing Real-Time Structured Analysis models. Proceedings of the XII Structure Methods Conference. Chicago, Illinois. Pages 301–313, Chicago, Structured Techniques Association, May 1987.

[Royce90] W. Royce. TRW's Ada Process Model for Incremental Development of Large Software Systems. 12th International Conference on Software Engineering, pages 2–11, 1990.

[TRW89] A. Marmor-Squires (principal investigator). Process Model for High Performance Trusted Systems in Ada. Technical Report, TRW Systems Division, Fairfax, Virginia, August 1989. 76 pages.

| | |
|---|---|
| [Ward&85] | P.T. Ward and S.J. Mellor. Structured Development for Real-Time Systems. Volume 1-3, Yourdon Press, New York, 1985-1986. |
| [Webb&86] | M. Webb, P. Ward. Executable Data Flow Diagrams: An Experimental Implementation. Proceedings of the Structured Development Forum VIII. Chicago, Illinois. Structured Techniques Association, pages 1–21, August 1986. |
| [Zultner89] | R. E. Zultner. Software Quality (Function) Deployment – Applying QFD to Software. 13th Rocky Mountain Quality Conference, 1989. Available from Zultner & Company, 12 Wallingford Drive, Princeton, NJ 08540. 11 pages. |