Deliverable D6.2

PECOS Project presentation

1. Identification

Project Id, Acronym:	IST-1999-20398 PECOS
Project name	Pervasive Component Systems
Deliverable Id:	D6.2, Project presentation
Date for delivery:	02.02.2001
Planned date for delivery:	31.10.2000
Classification	Public
WP(s) contributing to:	WP 6.2
Author(s):	Zeidler, Merker, Stelter – ABB Benedikt Schulz, FZI Stéphane Ducasse, UNIBE Fred Rivard, OTI

1.1 Abstract

This PECOS project presentation outlines motivation, goals and planed approach to realise component-based embedded systems. It documents the key issues, which are investigated, the major benefits expected and describe the technical approach anticipated at the beginning of the project.

The PECOS project aims to enable component-based software development for embedded systems. While focusing on architectural issues it touches upon the whole software development life cycle and addresses the major technological deficiencies of state-of-the-art component technology with respect to embedded systems by developing:

(i) a Component Model for embedded system components addressing behaviour specification and non-functional properties and constraints,
(ii) a Component Repository utilising this model, supporting a composition environment and interfacing to a component specification environment,
(iii) an interactive Composition Environment for composing embedded applications from components, validating functional (e.g., interfaces) and non-functional compositional constraints (e.g., power-consumption, code size), generating the application executable for the embedded device and monitoring their execution,
(iv) an Ultra-light Component Environment to install, run, test, and tune component-based applications on resource limited embedded systems and enable their management.

By providing a coherent approach and methodology for programming of component-based embedded systems PECOS enables an efficient and competitive embedded system development.

1.2 Keywords

Project goals, roadmap, and planed achievements

1.3 Version history

Ver	*Date*	*Editor(s)*	*Status & Notes*
1.0	22.11.00	Zeidler	First draft
1.1	13.12.00	Stelter	Review, Modification
1.2	20.12.00	Zeidler	Review and minor modifications
1.3	30.1.01	Zeidler	Incorporation of FZI comments
1.4	1.2.01	Zeidler	Incorporation of University of Berne comments

1.4 Classification

The classification of this document is done according to the security / dissemination level categories stated in Annex I (page 35) of the PECOS contract:

Classification	*Dissemination level*
Public (PU)	Public
Restricted (PP)	Restricted to other programme participants (including the Commission Services)
Restricted (RE)	Restricted to a group specified by the consortium (including the Commission Services)
Confidential (CO)	Confidential, only for members of the consortium (including the Commission Services)

1.5 Disclaimer

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

2. Table of Contents

3. Project presentation

3.1 List of Participants

ASEA BROWN BOVERI AG (ABB) ("the *coordinator*")
GOTTLIEB-DAIMLER-STRASSE 8, 69165, MANNHEIM, GERMANY
represented by its legal/statutory representative[s],
GERHARD ASSCHENFELDT, CONTROLLER and/or
DR. FRIEDRICH HARBACH, DEPARTMENT MANAGER
or his/her/their authorized representative[s],
DR. CHRISTIAN ZEIDLER – PROJECT MANANGER
PETER O. MÜLLER – TECHNICAL PROJECT MANAGER

UNIVERSITAET KARLSRUHE - FORSCHUNGSZENTRUM INFORMATIK (FZI)
HAID-UND-NEU STRASSE 10-14, 76131, KARLSRUHE, GERMANY
represented by its legal/statutory representative[s]
DIPL. WI.-ING. MICHAEL FLOR, CHIEF EXECUTIVE OFFICER
or his/her/their authorized representative[s],
BENEDIKT SCHULZ

OBJECT TECHNOLOGY INTERNATIONAL AG (OTI)
BURGEMEESTER HASPELSLAAN 131, 1181 NC, AMSTELVEEN, THE NETHERLANDS
represented by its legal/statutory representative[s],
DR. ERICH GAMMA, TECHNICAL AND LAB DIRECTOR, OTI AG (SWITZERLAND) and/or MR. ROBERT G. WHITE, CHIEF FINANCIAL OFFICER, OTI INC. (CANADA)
or his/her/their authorized representative[s],
DR. CHRIS LAFFRA

UNIVERSITAET BERN (UNIBE),
HOCHSSCHULSTRASSE 4, 3012, BERN, SWITZERLAND
represented by its legal/statutory representative[s],
PROF. DR. PETER MÜRNER, ACADEMIC DIRECTOR or
his/her/their authorized representative[s],
PROF. OSCAR NIERSTRASZ. DR. STÉPHANE DUCASSE, DR. ROEL WUYTS

3.2 Co-ordinator(s) Contact

Dr. Christian Zeidler,
Tel. : (+49 6221) 59-6259
Fax. : (+49 6221) 59-6253
email: Christian.Zeidler@de.abb.com
http://www.abb.com/decrc

Delivery Address
ABB AG, Corporate Research DECRC/A
P.O. Box 101332
D-69003 Heidelberg
Germany

Visiting Address
ABB AG, Corporate Research DECRC/A
Speyerer Strasse 4
D-69115 Heidelberg
Germany

3.3 Finances & Duration

Pecos project started at October 1st, 2000 and will terminate on September 30th, 2002. The following table illustrates the expenditures for each participant and the support granted by the European Commission. All figures represent expenses in Euro.

Participant Short Name	Total Costs	Adjusted Contribution from the Community
ABB	801602	400801
Co-ordination	145401	72700
Total Co-ordinator costs	947003	473501
FZI	467848	233923
OTI	764514	382257
UNIBE	302971	0
TOTAL	2482336	1089681

Table 1 Overview of the budget frame of the PECOS project

3.4 Objectives

3.4.1 Introduction

Component-based software engineering is quickly becoming a mainstream approach to software development. According to "Components, Objects and Development Environments: 1998, International Data Corporation" the expected turnover increase will be a factor of five from 1997 to 2002. At the same time there will be a massive shift from desktop applications to embedded systems. The PITAC report[1] describes this as the phenomenon of the "disappearing computer". More and more traditional IT systems will move from visible desktop computers to invisible embedded computers in intelligent apparatus. Furthermore, industrial automation systems become increasingly decentralised, relying on distributed embedded devices (intelligent field devices, smart sensors) to not only acquire but also pre-process data and run more and more sophisticated application programs (control functions, self-diagnostics, etc.). As a consequence of these facts, one can expect that component-based software engineering for embedded systems will be a key success factor for the European software industry in the coming decades.

But the state-of-the-art in software engineering for embedded systems is far behind other application areas. Software for embedded systems is typically monolithic and platform-dependent. These systems are hard to maintain, upgrade and customise, and they are almost impossible to port to other platforms. Component-based software engineering would bring a number of advantages to the embedded systems world such as fast development times, the ability to secure investments through re-use of existing components, and the ability for domain experts to interactively compose sophisticated embedded systems software. Visual techniques have been proven to be very effective in specific domains like GUI software composition. Composition of embedded systems software still has a long way to go to reach that level. At the very least, users would benefit greatly from the effective use of visual techniques for providing feedback in the development process (during design, composition, installation, and during runtime validation).

Unfortunately component-based software engineering cannot yet be easily applied to embedded systems development today for a number of reasons. Up to now, the mainstream IT players did not pay much attention to the (so far) relatively small embedded systems market and consequently did not provide it with suitable technologies or off-the-shelf software (such as operating systems). From a technical point of view, these choices were justified by considering the major characteristics of embedded devices, such as limited system resources (CPU power, memory, etc.) and man machine interface functionality, the typically harsh environmental conditions, and the fact that the development and target systems are not the same.

The rapidly growing market share of embedded systems is changing the equation and making investment in component-based software engineering for embedded systems not only viable but also essential. The key for European industries to benefit from the increasingly powerful and less expensive hardware, is the ability to develop and port embedded software more quickly and at acceptable costs. Vendors of embedded devices would

[1] The (US) presidential IT advisory committee report to the president "Information Technology Research: Investing in Our Future"

benefit by being able to offer scalable product families, whose functionality could be tailored by flexible composition of reusable building blocks. These families are differentiated by the performance of the hardware and the provided functionality, but are based on re-use of many identical software components. All this requires that the embedded systems software be modular and composed of loosely coupled, largely self-sufficient, and independently deployable software components.

The goal of PECOS is therefore to enable component-based software development of embedded systems by providing ***an environment that supports the specification, composition, configuration checking, and deployment of embedded systems*** built from software components.

3.4.2 Objectives

In order to reach its target the PECOS project must overcome the major obstacles to component-based software development for embedded systems:

I. Current approaches for component specification focus on interfaces only. *Behaviour and non-functional attributes (such as resource consumption and real-time requirements) are not captured.* Therefore we will define a **component model for embedded systems**, that captures and expresses such attributes for components and component architectures for embedded systems. The model will apply to or extend mainstream component models (COM+, JavaBeans/EJB, CORBA) and exploit existing specification standards and tools (e.g., UML, OCL, XML, IDL). Shortcomings of existing systems will be identified and solutions suggested.

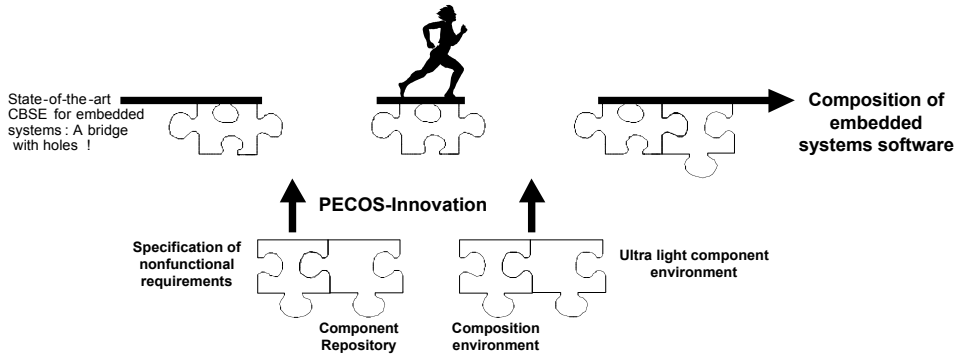

Figure 1 PECOS main objectives, closing the gaps in component-based software engineering

II. By the same token, *existing repositories neglect non-functional attributes and constraints*. We will therefore develop a **component repository,** which supports the representation, storage and retrieval of component specifications and implementations conforming to the component model for embedded systems. This achievement will be based as much as possible on existing technology, and therefore the main contribution lies in mapping the component model to available database or repository models.

III. Presently *we know of no suitable environments supporting interactive component composition for embedded systems.* Therefore we will develop a **composition environment** which supports graphical composition of software components for deployment to an ultra-light component environment. The key innovation of this environment will be its support for checking that component configurations meet their constraints, thereby improving reliability, robustness and overall quality while reducing development and maintenance time and costs.

IV. Currently, *no lightweight run-time environment for component-based embedded systems exist* and existing environments (WinCE+COM, embedded Java, embedded CORBA) do not address the requirements such as resource constraints and real-time execution for the majority of embedded systems. Therefore, PECOS will elaborate and implement an **ultra-light component environment**, to enable component-based services running on embedded systems. This component infrastructure will comprise support tools to manage, package, and distribute a component-based application on embedded devices, and optionally will enable interactive monitoring and management of the target applications.

3.4.3 Summary

The results of the PECOS project will enable the European embedded software industry to substantially improve software engineering of embedded systems by providing the missing technological links enabling usage of component-based software methods (see Figure 1). This brings about the ability to reuse components and component-based architectures in product families and the ability to build up a component market in the domain of embedded systems. This will in turn lead to improved productivity, decreased time to market, and increased competitiveness of the European industry.

3.5 Key issues

Component-based software engineering is of limited applicability to embedded systems development because presently available component technology fails to address several of the critical needs posed by these systems. The goal of PECOS is to overcome these obstacles through *technical innovations* that will enable the application of component technology to embedded systems, and *through process innovations* that will enable the transition from current approaches. These results will ultimately enable and lead to *product innovation* and originality.

The development of software for embedded devices requires that the behaviour of the software system and therefore of all its constituents be known precisely and unambiguously. Existing mainstream component technologies and methods lack the means to express behavioural contracts for components and to check contracts at design and run-time. Furthermore, the limited hardware resources of the target devices are inadequate for running the component infrastructure required for most component approaches. Again, existing component technologies (EJB, COM+, and CORBA) fall short in addressing this issue. While Figure 2 gives a high level view of the major PECOS topics and their relation to the different classes of innovation, the following tables (Table 2 to Table 5) describe how PECOS aims to progress beyond the state-of-the-art and thereby resolve the deficiencies described above.

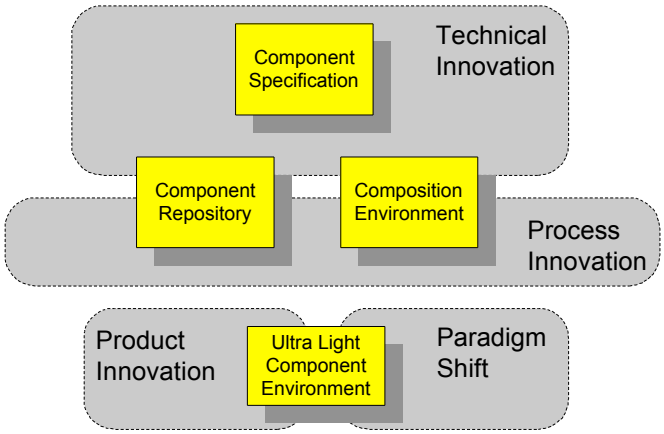

Figure 2 PECOS contribution to the different aspects of innovation

Topic: **Component Specification**
(S)tate-of-the-art and its (P)roblems: (S) Focus on syntactical interface description only (e.g. CORBA IDL). Behaviour, dependencies, and non-functional properties (e.g., time and space constraints) are not addressed. (S) Some research (trusted components initiative, design by contract) started to look at capturing such properties. But the link to embedded systems is missing so far, as well as the feasibility to apply the ideas to mainstream component technologies, and consequently neither IT (de facto) standards nor domain standards are established yet. (S) Configuration management systems describe dependencies between and different configurations of components. However they do not define a technique for component specification at all. (P) The lack of a sufficiently detailed description of software components prevents software developers of embedded devices from re-using existing code due to the potential risk of an ill-behaving component. Furthermore, tool and infrastructure supported checks are not possible.
Progress beyond the state-of-the-art: Proposition of an original component specification mechanism (contract expression) aimed at the area of embedded systems. This includes the definition and capturing of the concrete meta-data and relevant non-functional properties.
Evaluation of Innovation: High technical innovation. It is key to enabling many subsequent innovation steps that build on it.
(R)isks and risk (M)itigation: (R) Too much overhead and too complex for developer (M) Must be as invisible as possible, tools should be able to generate as much as possible automatically (R) Not accepted by standardisation bodies (M) Early involvement in standardisation work, use of already existing standards for component specification (IDL, RT-UML).

Table 2 Innovation of component specification

Topic: **Component Repository**
(S)tate-of-the-art and its (P)roblems: (S) Current repositories and subsequently their search and storage mechanisms are independent of software component specifications. (S) Component repositories with associated search facilities fall outside the scope of a development project and IDEs with integrated component repositories do not exist. (P) Tool supported code reuse (let alone software components) across project boundaries is hardly possible and support for product families is missing. (P) Company-wide (let alone world-wide) search for suitable software components is not straightforward.
Progress beyond the state-of-the-art: Component repository based on available database models (relational, object-oriented) that support the storage, search, and retrieval of software components considering the component contract specifications. This enables the sharing of components among various developers, and would allow tools (such as IDEs and design tools) to automatically search for relevant components given a set of contractual requirements.
Evaluation of Innovation: High technical and process innovation. The latter follows from the fact that make vs. buy decisions and the focus on integration will change the software development process. The proposed component repositories can also lead to innovative business models if component reuse across company border becomes possible and an embedded component market will be established (similar to today's ActiveX control market for the desktop).
(R)isks and risk (M)itigation: (R) General purpose facilities for search and reuse are not sufficient (M) focus on search and reuse facilities stemming from the actual case study and the domain of embedded systems

Table 3 Innovation of component repository

Topic: **Composition Environment**
(S)tate-of-the-art and its (P)roblems: (S) Graphical composition environments for component-based embedded systems software do not exist. (S) The existing development environments are specialised cross-platform development tools with different look and feel for the different platforms. (P) Development and test support is reduced to compiler and debugger functionality. (P) Testing is possible only on execution environment (target system) by low-level debug support.
Progress beyond the state-of-the-art: Graphical composition of embedded software components. Automatic composition constraint checking at design time. One composition environment for multiple target platforms.
Evaluation of Innovation: Very high technical innovation and process innovation because of its potential to drastically reduce development time and costs while improving quality.
(R)isks and risk (M)itigation: (R) Dependent on success of component specification (M) Tight coupling with the component specification work package, i.e., close co-operation and short iteration cycles between them. (R) Effort too demanding for PECOS project (M) Provide feasibility prototypes with reduced functionality but enough to attract EC companies to take up the idea and continue the challenge.

Table 4 Innovation of composition environment

Topic: **Ultra-light Component Environment**
(S)tate-of-the-art and its (P)roblems: (S) Plain C++/C-based target environments with HW dependent RTOS and APIs. (S) Tight integration and high dependencies between source code parts as well as between source code and RTOS, which causes side effects likely to appear when changing any parts (S) Debug and monitoring features are usually excluded from the final executable. (P) Parts of the source code or executable cannot be changed/upgraded individually. (P) Unwanted and costly side effects are likely to appear after any software modifications. (P) Target executable is hard to dynamically diagnose and monitor.
Progress beyond the state-of-the-art: Programming to a large extent target platform transparent. Support for dynamic component monitoring and reconfiguration. Seamless migration to the currently advancing real-time embedded JVM platforms. A component composition approach can be used at design time and the result compiled to a lightweight executable image.
Innovation: Technical innovation that leads to a very high product innovation (functionality, flexibility, and maintainability). It also has the potential to stimulate a paradigm shift in embedded system platforms.
(R)isks and risk (M)itigation: (R) Tool providers are not able to progress as expected (R) Target platform limitations are still too restrictive (M) component composition approach at design time but composition result compiled to lightweight executable.

Table 5 Innovation of ultra-light component environment

In summary, the innovation of the total project is far more than that of the sum of its parts. It is only through the integration of the individual results that the software development paradigm of embedded systems can be shifted towards components, software reuse is enabled, and component-based software may run on embedded systems.

3.6 Technical approach

The project work plan foresees a project duration of 24 months, lined up to focus on effective turnover of efforts spend towards productisation of results and gaining competitive technology advantages for EC companies through extensive exploitation. It is broken down to 6 work packages, where three of them *Requirements and Case Study*, *Dissemination and Exploitation* and *Management* are continuous activities throughout the project duration.

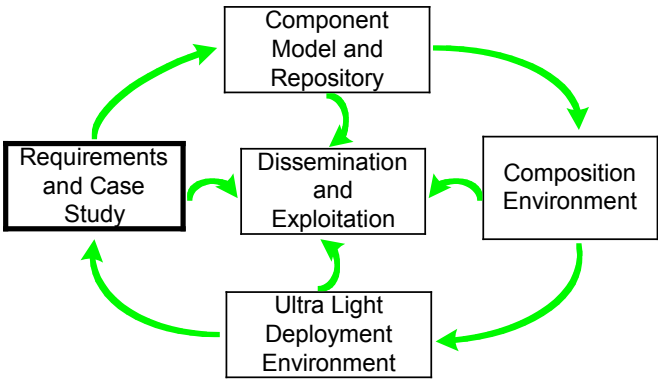

Figure 3 Work packages and their dependencies

Project phases are organised in an iterative and overlapping manner, compare Figure 3. Thus the dependencies of requirements are triggering the research on *Component Model and Repository*, and then trigger the implementation of *Composition Environment* and subsequently the *Ultra-light Component Environment* is created, which finally is validated by case study activities. The overlaps ensure that information flows from one package to another.

This is done for every major work package and leads to a focused sequential activity work flow to reach the goal of ***support for the specification, composition, configuration checking, and deployment of embedded systems component software***, as depicted in Figure 4.

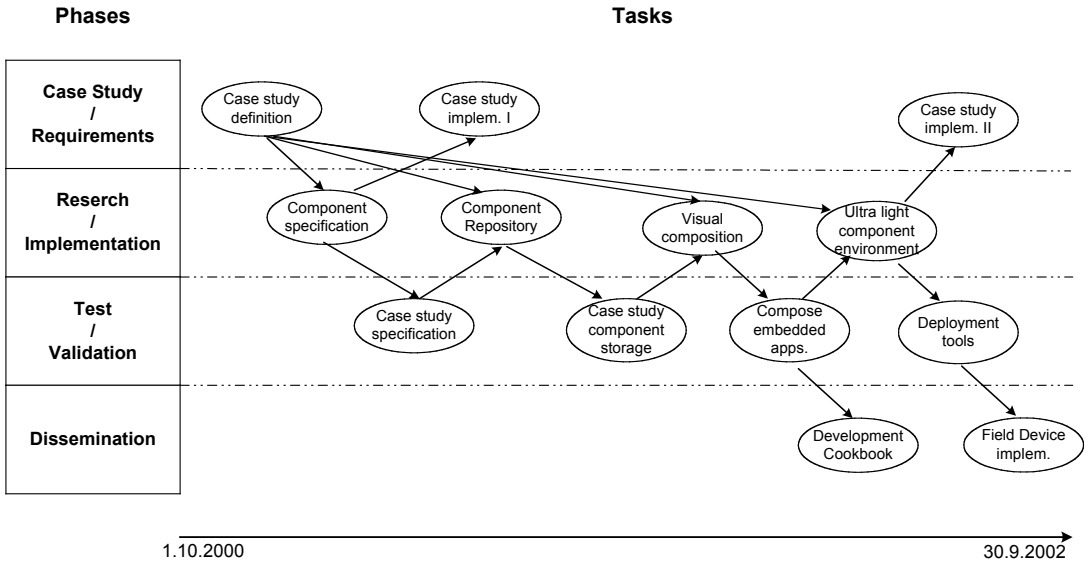

Figure 4 Iterative project execution workflow

In order to obtain results early and to productively benefit from early feedback, we are planning to use an iterative development cycle, which iterates over requirements, implementation and validation of interim results. Thus all activities are dependent on the case study definition, where the application scenario for embedded and ubiquitous computing software is defined. One the one hand, the case study provides the requirements for the development tasks (component specification, component repository, composition environment, ultra-light component environment). One the other hand, the results of these development tasks are integrated in an iterative way into the case study implementation for validation. Therefore, the case study implementation permanently feeds back to the concepts and tools developed so far. Finally the experiences of the complete endeavour are documented in the cookbook guidelines report.
Work packages are assigned according to partners expertise with thematic clustering. Thus the following set up of work packages represents the excellent way to ensure best results, since skills and interests match one to one.

3.6.1 WP 1: Requirements and Case Study (ABB)

The case study activities have three objectives:

- to deliver the requirements for component model, repository, composition environment, and ultra-light component environment,
- to assess and evaluate the results of the other work packages iteratively, and
- to support dissemination and exploitation of project results.

It is completely end user driven and takes care for industrial applicability of results.

The case study focuses on field devices such as temperature sensors, actuators, and other embedded equipment for industrial automation. Field devices impose hard restrictions on the available memory, processing, and communication resources due to the need of low hardware costs but also, even more important, due to the limited available power. For example, most field devices in the chemical industry have to work in intrinsically safe environments, which limits the available supply.
In this work package, standard component technologies and their available implementations like COM on Windows CE, CORBA on real-time OS, and embedded Java/JavaBeans are to be evaluated against the requirements of field devices. One can already foresee that the standard component technologies will not meet all requirements for field devices. On the other hand, field device firmware has a static configuration during run-time. There is no need to dynamically change the firmware by installing new components at run-time.
An approach to tackle the requirements for field devices could be to use component technology at design time to build the field device application and then to compile this application into a monolithic, optimised firmware. WP 4 (Ultra-light component environment) will address these issues.

For the prototype implementation, it is necessary to define a component-based field device architecture and to split today's field device functionality into potentially reusable components such as Fieldbus function blocks, parameter storage, alarm handling, sensor reading etc. In order to reach an acceptable level of component reusability, a range of field devices needs to be analysed. This analysis as basis for the component specification should include different device types like temperature sensors, pressure sensors, and actuators, but also different device types with respect to the communication (Fieldbus Foundation, Profibus, HART). One basis for the field device architecture will be the Fieldbus specifications.
From this analysis we gain a formal functional specification of the field device application and its components. In addition, we gain an informal non-functional specification such as real-time and synchronisation requirements. The informal non-functional specification provides important input for the component meta-model.

In order to assess and evaluate the results of the other work packages as soon as possible, the case study includes two iterations in the prototype implementation. The first iteration assesses WP 2 (Component Model & Repository). The specified components for one field device application are implemented using the component and architecture meta-model of WP 2. This implementation is done 'by hand' since no or only little tool support is available at this time. The first iteration includes actually two case study implementations: one C++/C-based and one Java-based implementation. One the one hand, these two implementations provide the requirements for WP 4 (Ultra-light component environment). On the other hand, two different implementation environments (especially the Java implementation) make sure, that the PECOS results are not limited only to field devices. An important issue of the Java-based case study is also to document how co-operation with a C/C++ runtime environment could be realised.
Basis for the C++/C-based case study will be an existing field device and its (monolithic) firmware of electric positioner for Profibus. Since the existing field device hardware will hardly be suitable for the Java-based case study, this case study will have a reduced functionality running on an evaluation-board (e.g. no Fieldbus and simulated process interface).

The second iteration assesses and evaluates WP 3 (Composition Environment) and WP 4. The field device application is re-implemented using the composition environment and the ultra-light component environment including the deployment tools.

Finally, the case study is used to prove the promises of component technology for embedded devices. While it is unrealistic to obtain reliable numbers about reduction of development and maintenance cost based on this case study, we are going to demonstrate the inherent flexibility of a component-based field device by:

- building a new field device version based on an existing design by adding/replacing components using the composition environment, and by
- implementing a customised version of a field device from the perspective of an OEM manufacturer.

This demonstration directly supports the dissemination of PECOS results.

3.6.2 WP 2: Component Model & Repository (UNIBE)

This task is concerned with developing (i) a component model for embedded system components, and (ii) a repository for storing, querying and exchanging descriptions of embedded system components.

The component model must build on and extend existing industrial component models by addressing the special needs of embedded systems. This work will focus on extending interface descriptions with contracts that express non-functional properties and constraints like resource consumption and real-time execution. The technical contributions will consist of a *component meta-model* for specifying properties and constraints of components relevant for embedded systems, and an *architectural meta-model* for expressing compositional properties of components. The latter activity will necessarily focus on capturing how compositions of components will affect critical properties of the resulting application, such as overall size, responsiveness, performance, throughput (in the case of transformational architectures), and run-time reconfigurability.

One of the key difficulties in developing the component model will be to decide *which* non-functional properties and constraints are most important to capture, and to develop a simple model that is general enough to capture a broad range of constraints. The requirements and validation criteria for the component model will therefore be determined primarily by the specific needs of the case study, and only secondarily by general considerations of embedded systems. Relevant ongoing efforts, such as pending OMG proposals to add real-time constraints to UML, will also be considered and incorporated, if appropriate.

The component repository will be the medium for storing specifications of components conforming to the component model. The key technical difficulties will be (i) mapping the component model to conventional database or repository models (e.g., OTI´s Envy), (ii) providing suitable query and navigation facilities, and (iii) interfacing to the composition environment.

The focus, however, will be on supporting the composition environment in *validating compositional constraints:* the repository must support application developers not only in finding components that implement needed functionality, but in directing them to sets of components that together will meet the non-functional constraints of the target device. Simple metrics and visualisation techniques will be adequate for addressing certain constraints, for example, application size, computational effort, and throughput can be measured, compared or combined in obvious ways that a repository can graphically present as part of a query interface.

3.6.3 WP 3: Composition Environment (FZI)

It is not enough to define components for embedded systems and to build up a repository only. It is also necessary to be able to plug them together efficiently. The composition of components should be graphically oriented to make use of the human ability to grasp structures presented in a graphical way easier than those presented in a textual form.

Today's composition environments do not support such visual composition techniques for applications of embedded systems. State of the market commercial composition environments (like Borland's JBuilder) are more or less focussed on GUI components and are not meant to deal with domain specific, non-GUI components, which are typical for embedded systems. Particularly, they do not deal with non-functional requirements and constraints mentioned above, which are of essential importance for embedded systems. Today's academic prototypes, on the other hand, which are mostly developed by the architectural systems research community, are not able to use component repositories and are able to handle only a limited set of non-functional requirements.

This work package aims to develop a graphical composition environment which is able to handle components that adhere to the specification and model defined in WP 2, allowing a graphical composition with respect to these constraints, and to efficiently use the component repository which is built up in WP 2.

The work package starts by collecting the requirements for such a composition environment from the analysis of the case study incorporating the expertise and experience of the user organisation. This task builds on the results of WP 2, to ensure that the composition environment can work with components that conform to the PECOS component and architecture models.
Based on the requirement analysis, a thorough examination of the state of the art will reveal the advantages and shortcomings of already existing tools and approaches. These results will be used as a foundation for the other tasks in this work package.

The next task defines the composition rules, i.e. the rules that specify how components can be put together to build a component based application. These composition rules should allow for the specification and verification of functional and non-functional constraints, context dependencies, and contracts, which are anchored in the component model and architecture.

Based on this work, we will develop a prototype of a graphical composition environment, which satisfies the special needs of component-based software engineering for embedded systems. The environment will provide general-purpose facilities for visualising, wiring, packaging and storing component compositions, retrieving components from the repository, as well as tailoring facilities to adapt the environment to different component sets and connectors. The environment will be responsible for code generation.

The focus of this work package is on composition rules and the actual composition of components. The visual aspect of the composition is an issue, but the result will not be a complete, stand-alone graphical IDE. We will rather try to re-use and interface to existing modelling environments or graphical editors whenever possible. The same holds for code generation: A full-fledged code generator is beyond the scope of the project, but if possible we will try to delegate code generation to existing tools (e.g. Rational Rose-RT or Rhapsody from ILogix).

3.6.4 WP 4: Ultra-light Component Environment (OTI)

Embedded devices impose severe restrictions on the available memory, processing, and communication resources (see WP 1). The component infrastructure intended to be deployed on embedded devices has to take these restrictions into account.

Therefore, we envision two options for the ultra-light component environment:

- a Java-based environment including a full range run-time component model, or
- a C++/C-based environment that utilises a component model only at design-time and a compilation of the designed application into a flat, monolithic firmware for run-time.

A full range run-time component model provides access to the embedded device's application at component level. Therefore, it supports:

- dynamic application reconfiguration by adding, removing, and upgrading components at run-time,
- enhanced debugging and tuning support by monitoring of components,
- high level support for distributed components.

Basis for the Java-based component environment will be OTI's VisualAge for Java Micro Edition.

The approach to compile a component-based application into a monolithic firmware (implemented in C/C++) makes component technology affordable for small embedded devices. The compilation is based on the component and architectural meta-model stored in the component repository and the application design obtained from the composition environment. One of the key difficulties is proper optimisation that trades in dynamic reconfiguration at run-time for small, low-resource applications. For example, dynamic event notification known from standard component models should be translated into direct function calls.

Both deployment options support packaging, versioning, downloading, and debugging support of component-based applications which were designed in the composition environment. For the application designer, the differences between the options should be as low as possible. By supporting both deployment options mentioned above in the ultra-light component environment, PECOS brings component technology into a wide range of embedded devices and provides a migration path towards Java-based embedded systems.

3.6.5 WP 5: Dissemination & Exploitation (all)

Dissemination and Exploitation focuses on two major activities. One on publication of theoretical results and second on demonstration of achievements through presentation of realised prototypes and demonstrators. Both activity types will be disseminated at well-recognised scientific conferences, journal and industrial fair as listed below:

- Disseminate the results of PECOS not only through the usual channels, such as conferences and articles, but also through actual products (tools as well as embedded devices) put on the (EC) market, thereby proving PECOS' applicability and usefulness.
- Publish results in various forms (journals, conferences, seminars, workshops, trade shows)
- Participation at industrial fairs, e.g., INTERKAMA, where we intend to present the results to a broad industrial audience
- Maintain a PECOS web-site
- Prepare the integration of the results in products (Visual Age for Java Micro Edition, field devices)

3.6.6 Milestones

Six major milestones are defined which represent the highlights of the project calendar, as depicted in Figure 5. They are composed of sub-milestones, which correspond to the work package list in section 3.4.2.

Consequently the following milestones reflect the major achievements of PECOS:
M1: Requirements of case-study (embedded devices for industrial automation)
M2: Component model (Meta model for component and architecture specification)
M3: Component repository prototype
M4: Composition environment prototype
M5: Ultra-light component environment (Java-based and C++/C-based prototypes)
M6: Embedded system prototype based on PECOS technology

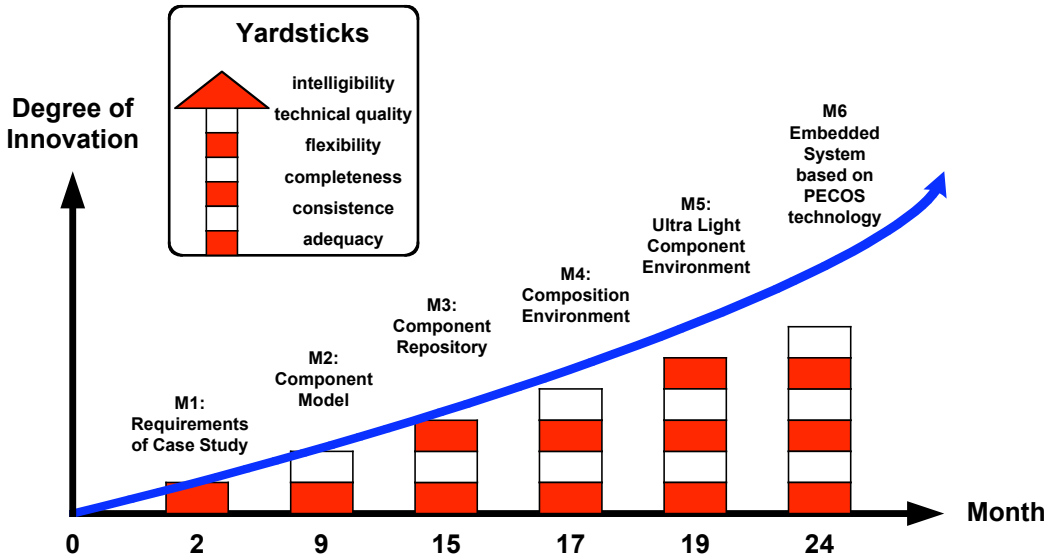

Figure 5 Major Milestones of PECOS Project

3.7 Expected achievements

3.7.1 Exploitation strategy

To extract maximum value from the results of PECOS, for both the PECOS partners and for European industry, an exploitation plan has been developed to encourage the take-up of component technology in Europe. ABB will analyse market of field devices during the project and provide aid to partners in adapting the exploitation strategy to the market situation at that time. Additionally, ABB will use the results of PECOS for

the development of component-based software for embedded systems at ABB's business units. The University of Berne and FZI will carry out the technology transfer. OTI will use the results of PECOS for improving tools in the area of application composition and its component models.

The transfer of the results of PECOS will be done principally through consulting services, through technology transfer projects, and development agreements with users of the software tools. OTI is currently in the process of developing a network of companies ("Partners") in Europe who are responsible for the transfer of embedded component-based Java technology to their customers. These Partners, to acquire the right to license VisualAge Micro Edition to their customers, must be fully trained in the technology, and provide training, technology transfer, custom engineering and support services to their customers. Agreements are currently being negotiated with MicroDoc (agreement is signed), ARXi, Turnkiek, IBM Germany, ENEA, and NSI (Israel). Additional Partners, with sufficient experience both in embedded software development and object-oriented technology are being sought after in France and the UK.

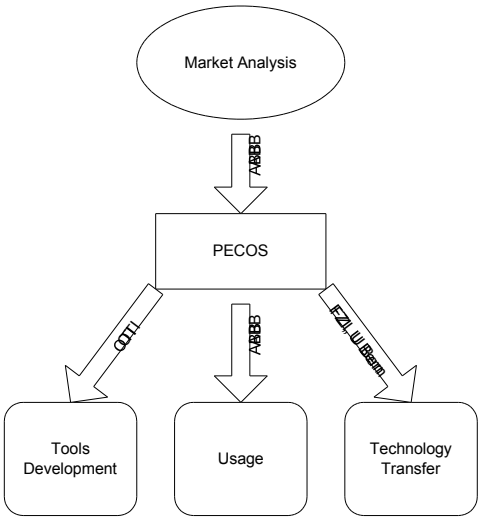

Figure 6 Exploitation strategy for PECOS

ABB intends to deploy the results of this project internally, developing their own component-based software for themselves and their customers. ABB expects to decrease the total cost of software development for embedded systems at ABB by 30% and to decrease the time to market by 40%.

One of the goals of the University of Berne and the main goal of the FZI is the technology transfer towards the European industry. Therefore they will exploit the results of this project by carrying through technology transfer projects concerned with component technology.

Partners	Target Group	Tools
FZI, University of Berne	Enterprises being interested in component technology or automation of CBSE	Carrying out best practice and technology transfer projects with industry partners.
OTI	Enterprises being interested in automation of CBSE	Selling add-on to various embedded software development tools.
OTI	Enterprises being interested in component technology	Setting up CBSE environment, producing and integrating of components.
ABB	ABB business units being interested in component technology	Using the methods and tools in ABB's projects.

Partner	Expected Advantages in 3 years
ABB	• More effective software development for ABB • Establish the reuse of components within ABB business units • Better ratio between costs and product features • Decrease effective programming and upgrade costs • Be prepared for potential embedded COTS market
FZI	• Improvement of visibility in industry and academia • Increased rate of projects with industry concerning component technology
OTI	• Increase in software development productivity due to rapid access to existing components. • Establishing a marketing channel, especially for SMEs. • As components become more widely used, specialisation will occur where more and more components will be developed by experts in the field. • Increased ability to compete with Asian firms in the development and deployment of devices employing embedded processors.
University of Berne	• Improvement of the visibility of lab on the international research community • Improvement of the teaching curriculum

3.8 Scientific and technological European prospects and dissemination strategy

The dissemination of the PECOS technology, and indeed embedded Java technology in general, will be a challenge. The embedded software industry has been slow at adopting advanced software paradigms, such as object-oriented technology, let alone pervasive component systems. No company, not even IBM, has the resources to disseminate, effectively, the technology to this industry on its own. To increase the "technology transfer bandwidth", as mentioned in the Section 8.1, OTI is establishing a network of European Partners who will be highly trained in the embedded Java technology, and will therefore be primed to handle the component system technology as it becomes available. These Partners are being chosen for their expertise in OO technology, embedded technology experience, and their marketing channels in a particular industry sector.

The FZI and the University of Berne have a lot of experience and great interest in scientific publications. Also, FZI and the University of Berne have been involved in organising various conference events (workshops, tutorials) and as such have influenced the scientific community. There are several employees of FZI and University of Berne who are working on their PhD thesis concerned with component technology. Both FZI and University of Berne are giving lectures for students in computer science, engineering, and related courses (the FZI in assistance to the University of Karlsruhe). Therefore FZI and University of Berne will be responsible for dissemination in the scientific community and the next professional generation.

ABB is a direct user of the technology resulting from this project. Due to the fact that this technology is especially useful for the support of the co-operation of different enterprises, the dissemination of the result of this project to ABB's partners in other projects is of high value. Therefore ABB will make large efforts to convince its future project partners to also use the new technology.

Altogether dissemination will be done to the scientific and industrial community as well as to the next generation professionals with the help of scientific and industrial conferences, books, advertisements, industrial co-operation and lectures at universities. This will increase the maturity of European software engineers and thus increase the competitiveness of European industry.

Partners	Target Group	Tools
FZI, University of Berne	Scientific Community in Component Technology	Scientific Conferences, magazines
FZI, University of Berne	Students in computer science, engineering, and related courses	Lectures at the Universities of Karlsruhe and Berne
OTI	Enterprises being interested in automation of CBSE	Advertisements for industry-oriented conferences, books, workshops.
OTI	Enterprises being interested in Component technology.	Partnerships, workshops, seminars, talks, www, commercial advertising. In-house seminars and workshops for IBM.
ABB	Enterprises being interested of CBSE in the field of automation technology	Advertisements for consultancy activities, industry-oriented conferences
ABB	ABB business units interested in component technology	Industrial co-operation, consultancy by ABB Corporate Research