Deliverable D2.1

Requirements specification for embedded systems component

1. Identification

Project Id:	IST-1999-20398 PECOS
Deliverable Id:	D2.1, Requirements specification for embedded systems component
Date for delivery:	15 January 2000
Planned date for delivery:	1 January 2000
Classification	Public
WP(s) contributing to:	WP 2
Author(s):	Stéphane Ducasse, Roel Wuyts, Gabriela Arevalo University of Bern

1.1 Abstract

This document describes the requirements for the meta-model. It serves as the base for the meta-model deliverable (D2.2.1). It studies the requirements for the meta-model and the composition described in D1.1, and the detailed required for the field-device as discussed in the 'Field Device Workshop' in Minden. Then it introduces the iterative approach that is going to be used to mine more and better requirements using a prototyping approach.

1.2 Keywords

Meta-model, component, composition, architectural style, requirements

1.3 Version history

Ver	*Date*	*Editor(s)*	*Status & Notes*
1.0	11.02.01	Stéphane Ducasse, Roel Wuyts, Gabriela Arevalo	First draft
1.1	12.03.01	Stéphane Ducasse, Roel Wuyts, Gabriela Arevalo	Comments from ABB included.

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

2. Table of Contents

3. List of Achievements

This deliverable consists of the following achievements:

Report Title / Document Id	Contents
D2.1 Requirements spcification for embedded systems component	Deliverable D2.1

4. Introduction

As the current document serves as input only for the component meta-model (D2.1.1) and that the document D2.1 addresses the development process aspects, the current document presents the analysis of the requirements with the particular view of component meta-model and component composition. This document first extracts meta-model requirements from the document D1.1, then presents the results of the workshop held at Minden and finally explains how the prototype we are building allow us to have concrete feedback on the component meta-model. With this prototype we gain an understanding on the notion of component properties validation and composition.

5. Meta-Model Requirement Analysis

The meta-model requirements have been gathered from two sources: the document D1.1, the Field Device Workshop held at Minden and from the prototype we are currently building. Using this prototype the presented requirements will be extended later on. The following three sections discuss the requirement analysis from D1.1, the first concrete requirements from the Field Device Workshop and the prototype approach.

5.1 D1.1 Requirement Analysis

Here is the list of requirements that have to be taken into account to design, the component meta-model and the composition architecture. The component meta-model should allow the expression of the following requirements.

5.1.1 Component Meta Model

Non-functional Properties

The component meta model has to provide concepts to model non-functional attributes. Non-functional attributes are e.g. memory consumption, execution time etc.

Engineering Tags

It must be possible to define tags that are used for deriving engineering data (□ DTM, FDT).

Design by Contract

The component meta model has to provide concepts to model invariants (pre- and post conditions) on components and especially their interfaces. It must be possible to check the fulfillment of these invariants automatically.

Dependencies

Dependencies of a component from other components must be able to model. Especially the dependencies from certain interfaces or run-time services must be able to model. There should be a way to describe the needs and required services of a component. The question is whether the RTOS services (□ e.g. POSIX) are part of the component model so that they can be specified as required.

Extensibility

The property and contract concept is pushed by a predefined set of properties and contracts. Still it must be extensible so that further categories of properties and contracts can be added.

Classification

A component can be classified. This affects e.g. a component's visibility, which could be restricted to a project, group, department etc.

Interfaces

The component meta model has to provide concepts to specify a component's interfaces. These concepts shall be as flexible as possible to allow e.g. procedural and object-oriented (with and without polymorphism) interfaces.

Type safety must be guaranteed. The interface specification has to be separated from the implementation.

Multiple Interfaces

It must be possible to define more than one interface for a component.

Unique Identifier

Each component has to have a unique ID by which it can always be identified.

Key Words

Each component is characterized by a few key words (the abstract).

Containment

The model has to support building hierarchies.

Completeness

The model and its contents must be sufficient (contain enough information) for code generation.

Portability

The component meta model has to be implementation language independent. However, the concepts must be realizable at least in C/C++ and Java.

Textual Format / Openness

The specification of components has to be in a textual format. The format has to be tool and vendor independent (□ use open standards).

5.1.2 Composition model

Architectural Constraints

The composition model has to provide concepts for modeling constraints on architectural level. Constraints for e.g. interface type compliance between components, overall memory constraint fulfillment etc. It must be possible to check the fulfillment of constraints automatically (validation rules).

Composition Rules

Concepts for the specification of at least two categories of composition rules have to be provided. These are computation and validation rules.

Computation rules are used to derive non-functional properties of composites from their underlying components.

Validation rules are used to validate a composition based on its non-functional properties and constraints.

Textual Format / Openness

The specification of composition rules and architectural constraints has to be in a textual format. The format has to be tool and vendor independent (□ use open standards).

5.2 Field Device Meta-Model Requirements

The brainstorming actions held during the kick-off meeting and the Field Device Workshop resulted in the following view on components for embedded devices:
A component should have:

- A unique identifier
- A documentation
- Ports (input/output)
- Contracts
- Property set
- Possible required components

Of course, components also need to conform to certain consistency and composition rules. In order to obtain a useful meta-model, it is important to have a concrete understanding of such rules. These rules can only be given by the domain experts, but it is always very hard to extract them. Therefore, in Minden we obtained a first list of rules components have to comply to. This list will later on be extended (see 5.3 where we discuss the prototype approach):

1. The component can be *active* or *passive.*
2. The component considered as *active* must have an scheduler.
3. If a component is *active*, and has a scheduler then one of its properties must be the *cycle time.*
4. If a component is *passive*, then it is necessary to have a scheduler or just another component that is *active.*
5. If two components must interact, then synchronization is needed.
6. According to the Software Requirements Doc, and FBlock has input/output ports and contained parameters, then the contained parameters can only be contained in FBlocks and in Transducer Blocks.
7. The *executable time* is a property of a components and also of the methods.
8. (Obvious) The total time of the executable methods time must be less that the specified scheduler time.
9. Only the FBlocks fetch data.
10. A container must at least have a FBlock and a Transducer Block.
11. The component types has a specified (minimal) set of parameters.
12. Cyclic Data Frame can only contain parameters of the Type In.

5.3 Prototype approach to gather more requirements

Because it is always difficult to extract domain , it was decided at the Field Device workshop to follow a prototyping approach. In such an approach, the idea is to gather an initial set of requirements, and build a prototype that supports these requirements. Then, together with the domain experts, more requirements are harvested, and the prototype is updated to take these into account. This is repeated until the domain rules are expressed satisfactory to the experts. The output is thus two-fold: first, a concrete set of requirements and second, a prototype supporting these requirements.

We are currently prototyping a component framework using the object-oriented language Smalltalk. Smalltalk was chosen because it is a very well suited environment for rapid prototyping, and because its reflective facilities allows us flexibility and a fast feedback loop. The meta-model we are currently implementing is very flexible and extensible, and can be customized to suit the requirements from previous sections. In the prototype, a component has:

- An unique identifier
- A list of properties
- A set of associated rules that support composition, computation or validation of the properties
- Validation rules, computation rules and composition rules should be expressed and associated with a component, a set of components, or its properties.

This model supports components as described in Section 5.2, and we are currently expressing the validation rules. As said before, once it is finished, this prototype will then be used to gather more concrete requirements.

6. Conclusion

In this document we list the view on components and their validation and composition rules that was discussed in the Field Device Workshop at Minden. We introduce the prototype approach that we will use to gain more insight in the problem domain, and support with a meta-model.