Market Enabler for Retarget-able COTS Components in Embedded Domain

Philippe Robin, Sylvain Robert, Michel Sall, Sébastien Berthier

Market Enabler for Retarget-able COTS Components in Embedded Domain

Philippe Robin[1], Sylvain Robert[2], Michel Sall[1], Sébastien Berthier[1]

1: Trialog, 25 rue du General Foy, 75008 Paris
philippe.robin@trialog.com
2: CEA-List, DRT/LIST/DTSI/SOL, CEA Saclay 91191, Gif sur Yvette Cedex
sylvain.robert@cea.fr

Abstract: A general trend in real-time and embedded (RTE) domain is towards complexity. Large, highly integrated, and functionally extensive systems are developed, with an invariant need for costs and time-to-market mitigation. This tendency has set a need for overhauling development methods in the RTE area. Using software Components Off-The-Shelves (COTS) could be an answer to this concern. Effectively, COTS are cheaper, since they are reused several times. They are also safer, since they have been used and validated extensively by different users. However, several major obstacles prevent software COTS usage from spreading in RTE systems development. First, the heterogeneity of components forbids seamless integration. Then, the poorness of components characterization makes it difficult to choose the right component. Last but not least, the absence of COTS usage support tools and methodological guidelines discourages many developers. MERCED (Market Enabler for Retarget-able COTS components in Embedded Domain) is a European ITEA research project, which aims at providing solutions to these concerns, and this way, giving an impulse to the emergence of a European software COTS market. This paper outlines the main technical contents of the MERCED project, with a particular focus on the execution framework on top of OSEK, and describes an industrial use case of this technology.

Keywords: Components, Off-The-Shelves, Real Time Embedded, Reuse, CCM, OSEK.

1. Introduction

The paper structure is as follows: section 2 describes MERCED project contents. Then, section 3 focuses on Real-Time and embedded concerns.

2. The MERCED approach

2.1 Components Reuse Environment for real-time and embedded systems

The first goal of the MERCED project is to offer a components reuse environment which will provide a support for components storage & classification on the one hand, and components selection and retrieval on the other hand. In the proposed scheme, two actors are distinguished. The first is the components developer who feeds the components repository with his products. He has to provide a description with each component. This description has to be compliant with the MERCED RTE components taxonomy. Then, the second is the components user, who needs to retrieve specific components. The MERCED environment offers tools to support the user in his search (e.g. search engine), as well as in the following steps: component validation, simulation, and testing.

Components classification and retrieval

MERCED will provide a component meta-model that gives a common and unified component specification. This meta-model will help automating many activities related with searching, retrieving, qualifying, testing and integration, since prior to usage it is already known which specification information can be manipulated. The meta-model will be used to characterize components within the reuse environment. Doing so, the information required for selecting and using components will be available to search tools. The meta-model will include extendable taxonomy means for testing concepts (mockup environment, test suite definition, test driver components, stub components, qualification level, etc). Then, a suitable approach for component documentation and searching with extendibility

capability will also be provided. The initial taxonomy for non functional and functional properties will be defined in such a way that it can easily evolve overtime: proper taxonomy extendibility mechanisms are thus to be defined and provided.

Component validation and qualification tools

To make the embedded real time software development teams confident in reusing components, these components must reach the qualification level they need. This qualification level is project-dependent and can be one of the components characteristics. For example, an avionic integrator will need that a component must be certified at the "DO-178B level A" level to be integrated into some critical subsystem. This qualification level can be part of the component description. Moreover, in the component selection process, the potential reuser has to ensure himself that the selected component description matches his needs, and will then also need to be confident in the fact that the component is conform to its description.

To achieve component qualification, a test platform at the hardware and OS abstraction level will be provided by MERCED. The test platform will allow defining and possibly running the component test suites in a virtual environment simulation.

System simulation tools

The selection of an existing component to be integrated in the designed system lays on a "search, select and try" process. The "search" and "select" steps are achieved by the presence of a description of each existing component in the repository and the associated searching and filtering tools based on this description.

The "try" step, consists of quickly verifying that component interacts as expected inside the system. Powerful means for the inspection of the component interactions inside the system will be provided in a simulation environment providing traces, break conditions, system state inspection, and other debugging features that will not be available on the final target or induce heavy supplementary overhead.

2.2 The MERCED Execution Platform for Real-Time and Embedded systems

The second purpose of MERCED is to develop an execution framework for RTE systems. The base technology chosen for this framework is the CORBA component Model in its lightweight version [2].

This latter enforces an original execution model, based on a strict separation of concerns between business and non-functional aspects. All business logic is gathered in applicative components, while all non-functional processing is embedded into containers. Containers provide an execution context to components: they mediate all interactions to/from the external environment and doing so, shield components from any execution platform-dependent issues.

The MERCED framework itself is mainly made of the following parts (Figure 1):

- the execution infrastructure, comprising technical services
- the containers to fill the gap between the components and the execution infrastructure
- the administration service allowing assembly, configuration and deployment of the components and their containers
- a tool allowing to build the containers according to the application requirements
- some predefined components, providing generic services

The main purpose of this part of the project will be to realize a reference implementation of this framework on top of Real Time CORBA [2], according to the definitions and requirements issued from users, which are gathered and analyzed also in the context of the project. The aim is to design a retarget-able framework, i.e. a framework easily portable to another execution infrastructure. In order to demonstrate this ability, a porting will be performed during the project.

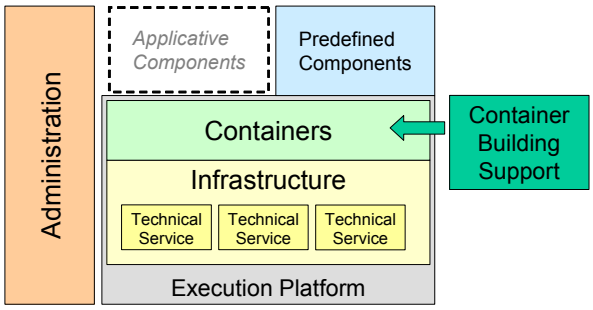

Figure 1 : The MERCED Execution and Administration framework

The work related to framework implantation will more precisely involve the following tasks:

- Interaction models
- Container runtime reference implementation
- Container Building Support
- Predefined components implementation
- Configuration and Deployment Suite
- Port of the Execution Support on top of OSEK

Interaction models

In its original shape, the CORBA component model offers limited interaction support. In the context of real time and embedded systems, it is necessary to make available an extensive range of highly customizable interaction mechanisms, with e.g., QoS enforcement. This task thus aims at adapting accordingly the CCM. A significant amount of work has already been performed on this task, and some results are available. The core idea resides in adding "connectors" to the CCM. These new elements are software entities dedicated to interaction management [3], which are used to interconnect components. All necessary modifications of the CCM have already been performed to include this new software entity, and a large part of prototyping has been performed. More details on this topic can be found in [4].

Container runtime reference implementation

The purpose of this task is to specify and implement the container runtime so that the CCM containers can integrate nicely all the technical properties that are to be provided to the components. It should be noted that some parts of the containers implementation may depend on the underlying execution platform. The starting point is the Lightweight CCM OMG specification, and the reference implementation is Real Time CORBA based. Guidelines for porting on another infrastructure will also be provided.

Container Building Support

In the CCM approach, containers are generated. First, a description of components, is written in a dedicated language, namely the Interface Definition Language (IDL). Then this description is used as input to a generation tool which produces the container code. Since the MERCED project will largely impact the contents of containers (in order to specialize them for real time support), it is necessary to provide the corresponding generation tools. Note that the generation tool is also part of the retarget-able environment. A first version will be provided to generate Real Time CORBA compliant container, but porting rules will also be issued, and a porting will be performed.

Predefined components implementation

The framework will be completed by predefined components; these latter are components that contain processing which is at the same time specific (to a given application domain, e.g. telecom) and generic (the processing is of interest for all applications of the given domain). These predefined components will be implemented for two application domains: software radio and electrical distribution. It should be noted that as these components are, by construction, plain components, they will benefit, as any application component, from the hosting support provided by the execution platform and that therefore their integration will be obvious.

Configuration and Deployment Suite

The CCM is accompanied by another OMG specification, which provides guidance in application assembly and deployment [5]. The "Assembly" is a step in which the application architecture is described, in terms of components instances, and interconnections. This description is based on a dedicated (XML) format, also defined in the specification. Once the assembly is done, a deployment tool is fed with the architecture description, and it performs accordingly components instantiations and interconnections. All these facilities will be adapted in the scope of the MERCED project in order to fulfil RTE-specific requirements, and then will be implemented. A particular attention will be paid to the embed-ability of the deployment tool.

Port of the Execution Support on top of OSEK

As said before, our aim is to design an easily portable framework. In order for us to assess the retarget-ability of the MERCED approach, a porting of an OSEK framework has been realised to a micro controller well-known in the embedded world together with the porting of the CCM framework on top of this OSEK framework. This is discussed in the following section.

3. CCM framework in the OSEK environment

OSEK is an execution infrastructure widely used in the automotive industry. Being an ISO standard it attracts a growing number of other industrial fields. Therefore it is a good candidate for ERTS projects

3.1 The OSEK environment

OSEK is specialised for highly constrained hardware platforms. It is mainly composed of two parts : an operating system (OSEK-OS [6]) coupled with a communication environment (OSEK-COM [7]).

Operating system support offered by OSEK

OSEK-OS is a multi-tasking operating system. It is highly static: all parts that can be stored in ROM are stored there to minimise the amount of required RAM. The declaration of tasks, events, counters, alarms, resources are defined in a static

configuration, and so are tasks and resources priorities.

Communication infrastructure under OSEK

The OSEK-COM specification defines a simple message-based communication means. Messages are defined statically, meaning that their sender, receivers and maximum size are defined statically. The message content is defined by means of a so-called *message object*. There are two types of Message Objects, a static with a fixed structure and a constant length and a dynamic whose length is variable.

OSEK implementation

OSEK-OS has been ported on an ARM7 microcontroller with 256 Kb ROM and 64 Kb RAM.OSEK-OS occupies 6% of the ROM and 0.3% of the RAM. Some trials have shown that typically the library (lib C/C++) needed by the components occupies about 17% of the ROM and 12% of the RAM. Integration on OSEK-COM is under way.

3.2 Component localisation and communications

Component localisation

The different components of an application can be localised on the same CPU (see figure 2) or distributed on several CPUs (see figure 3).

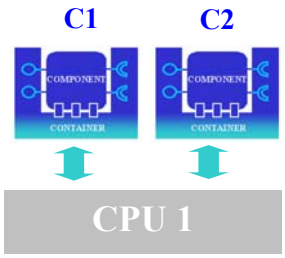

Figure 2 : All the components (e.g. C1 and C2) of the application are running on a single CPU

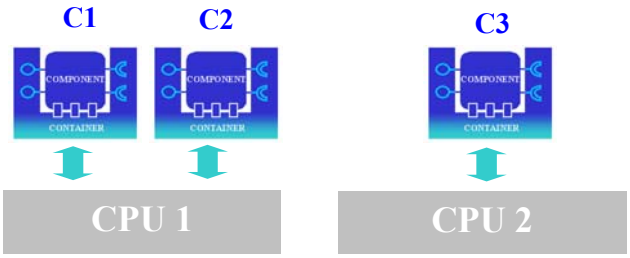

Figure 3 : The components (e.g. C1, C2 and C3) of the application are distributed on several CPUs

Inter components communication and execution

The figure 4 shows the interaction between components from the business developer point of view.

Figure 4 : Interaction between components from component point of view.

But in reality, this interaction is achieved through the use of containers which mask the mechanisms used to perform it (see Figure 5). This masking allows components to be unaware of whether the call is local or remote. It also lets them be independent of the actual mechanisms used for the interaction.

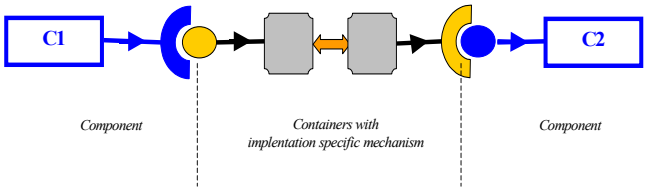

Figure 5 : Interaction between components as it is in reality.

Suppose that C1 wants to call a function implemented in C2 (see figure 4). This call is intercepted by the container associated to C1. This container interacts with the container associated to C2 through a mechanism described below. And finally, this container calls the targeted function of C2.

Three mechanisms have been found necessary and sufficient for inter components communication and execution in the context of OSEK. :

- synchronous call
- asynchronous call
- periodic activation.

For local interactions, these mechanisms, depicted in figure 6 below are.

- **synchronous call :** The container of C1 directly calls the container of C2
- **asynchronous call** : the container of C1 sends an event to the container of C2
- **periodic activation :** this mechanism is achieved through the use of alarms and counters together in interaction with a timer component which is part of the framework.

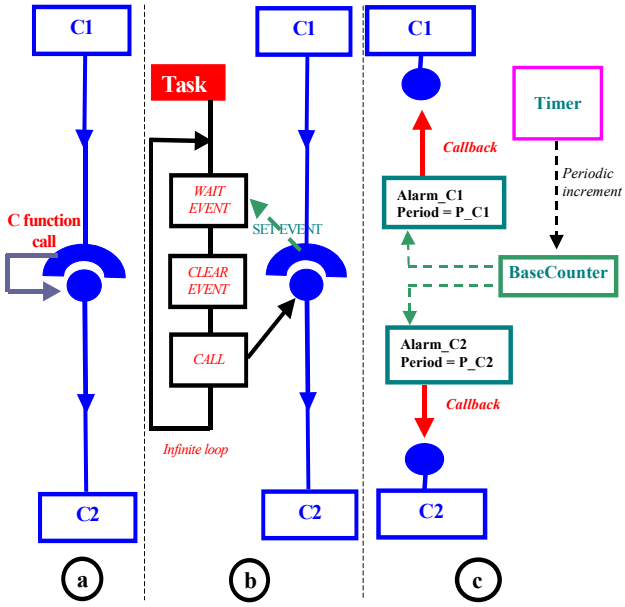

Figure 6 : Synchronous call (a), asynchronous call (b), periodic activation (c) for local calls.

Note : In order to simplify the drawing, containers of figure 5 have not been shown on figure 6.

For remote calls, synchronism and asynchronism will be achieved through the exchange of messages via OSEK-COM as depicted on figure 7.

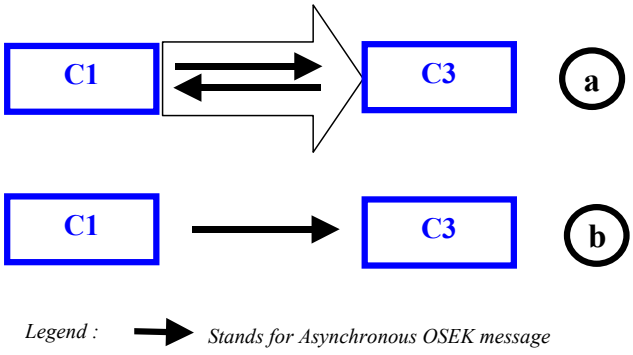

Figure 7 : Remote synchronous call (a), remote asynchronous call (b).

3.3 OSEK and CCM generation chain

The application architect searches for existing COTS using MERCED search tool and looks for application specific components when no suitable COTS exists (each component is delivered with its IDL description). Then he specifies the architecture deployment which details the components layout on CPUs.

Generations tools are used to generate :

- a makefile
- OIL description files (one per CPU) [8]
- Container sources

and to select components (objects or libraries for COTS, sources for application specific ones).

Those files are compiled and linked to build the application executables (one per CPU). The complete chain is described on Figure 8.

During this process, the only manual operations performed are:

- searching, selecting and trying the required COTS,
- eventually developing the missing components,
- defining the deployment architecture.

All other operations are automatically executed by tools developed during the MERCED project. These tools are part of the CCM and OSEK frameworks.

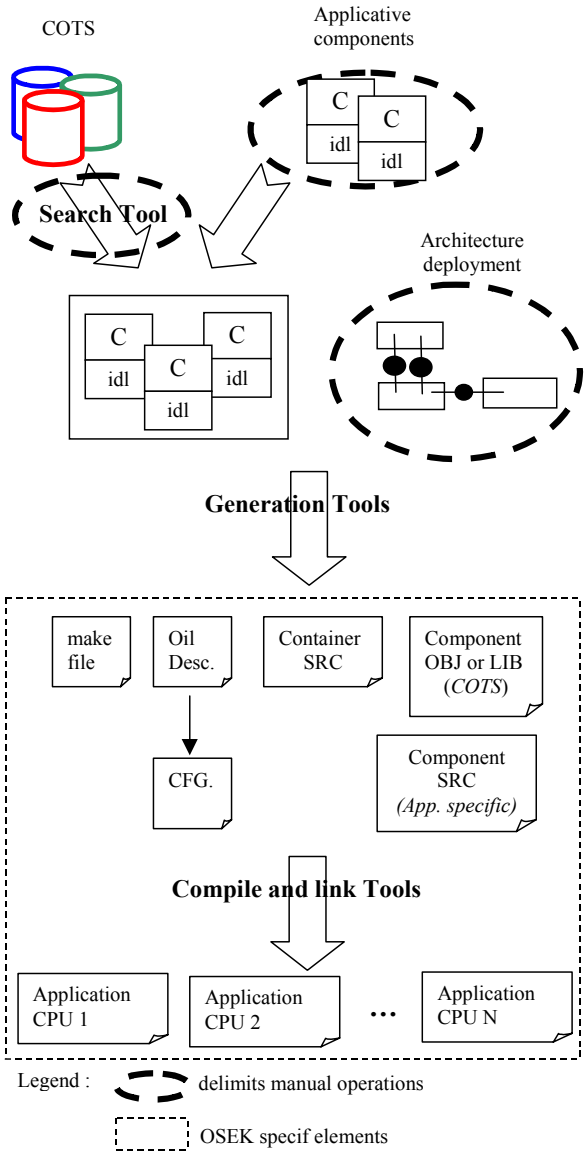

Figure 8 : The complete generation chain

3.4 Industrial use cases

Use cases are a way for the Merced project to test the solutions and concepts proposed by the consortium. Altogether they hightlight the following issues :

- create a component in compliance with the "Merced Component Model";
- publish a component so that it will catch the interest of a potential customer;
- download a component and integrate it in an execution environment providing services as required by the component description;
- assemble and deploy components;
- generate and distribute an application over a set of CPUs connected through a communication layer.

One of these use cases is implemented on top of OSEK for the electrical distribution.

4. Conclusion

In this paper, we sketched the main contents of the MERCED European project. This project proposes to compensate the rise in complexity in the RTE domain by enforcing a COTS-based development approach which would permit to decrease development costs while keeping a sufficient level of reliability. In order for COTS to achieve their breakthrough in the embedded system area, proper tools and methods have been defined by MERCED.

The main difficulty the project had to deal with was to select and adapt a subset of the MERCED lightweight CCM to static platforms like OSEK.

Three partners from the electrical distribution, software radio, metal processing domains are currently validating this approach.

5. Acknowledgement

The authors acknowledge the contribution of their colleagues to this work.

The comsortium is made of the following companies and institutes : Axlog , CEA-LIST , Schneider Electric, THALES Communications TIMA-INPG ; Trialog (France) ; ENEA (Sweden) and DS2, European Software Institute, MSI, TELVENT, UPM (Spain)

6. References

[1] Object Management Group, Lightweight CORBA Component Model - OMG draft adopted specification, 2003.

[2] Object Management Group, Real-Time CORBA specification version 2.0, formal/03-11-01, 2003.

[3] D. Balek and F. Plasil, Software connectors and their roles in component development, DAIS'2001.

[4] S. Robert, A. Radermacher, V. Seignole et al., How ADLs can help in adapting the CORBA Component Model to Real-Time Embedded Software Design, in From Specification to Embedded Systems Application, pp. 117-132, Springer, 2004.

[5] Object Management Group, Specification for deployment and configuration of component-based applications - draft adopted specification, 2003.

[6] OSEK/VDX, OSEK/VDX Operating System version 2.2.3, 2005.

[7] OSEK/VDX, OSEK/VDX Communication version 3.0.3, 2004.

[8] OSEK/VDX, OIL: OSEK Implementation Language version 2.5, 2004.

7. Glossary

ERTS Embedded Real Time Software

CCM: CORBA Component Model

RTE: Real-Time Embedded