Information Societies Technology (IST)

Advanced Research and Technology for Embedded Intelligence in Systems

WORKING DOCUMENT

Seamless Connectivity and Middleware

-

Priorities Analysis

Editor: Christophe Lécluse
Rudy Lauwereins

Status - Version: First Draft
Date: 7/7/2006
Confidentiality Level: Confidential

Acknowledgements

The editors acknowledge contributions by all the other contributors.

Document revision history

Date	Version	Editor/Contributor	Comments
7/7/2006	1.0		First draft – Results from the ARTEMIS Summer camp
18/7/2006	1.1	Alun Foster	Incorporation of first feedback and text augmentation

Table of Contents

1 INTRODUCTION

The Strategic Research Agenda (SRA) developed, though open consultation, by the members of various ARTEMIS working groups, contains an extensive list of research topics that are felt to be the most important issues to be tackled in the short to medium term. As a refinement that will help guide proposers of projects in the short term that wish to refer to the ARTEMIS SRA, ARTEMIS members delegated experts to an extensive working session, during which the priorities for the short term were put forward and discussed for each of the major industrial SRA domains. The results of this workshop are documented in this and two accompanying documents.

The research topics identified as most urgent may be taken as a guide when proposing research projects for execution under the Framework Programme 7 of the European Commission, under the EURKA clusters ITEA-2 or MEDEA+, or indeed under locally focussed research projects. In all cases, the results will contribute to the long-term objectives of ARTEMIS.

2 RANKING OF THE PRIORITIES

The priorities have been ranked by the organizations represented at the Summer Camp as follows :

SC&M

❖Resource Management	43
❖Robustness & diagnosis	34
❖Programming	30
❖Organization & deployment	30
❖Provably correct systems	23
❖Global connectivity	22
❖Security	6
❖Data distribution	2

Priorities have been split into three categories:

1. Highest priority
    - Resource management
2. High priority
    - Robustness & diagnosis
    - Programming
    - Organization & deployment
    - Provably correct systems
    - Global connectivity
3. Medium priorities
    - Security
    - Data distribution

3 APPLICATION DOMAIN CLUSTERS ANALYSIS

Assigning voting companies to three different application domain clusters, according to the SRA classification, we get some more application domain specific prioritization that is the following.

3.1 Critical cluster

The four top priorities, in decreasing order of priorities are :

1. Provably correct systems
2. Robustness & diagnosis
3. Resource management
4. Programming

3.2 Device & plant cluster

The four top priorities, in decreasing order of priorities are :

1. Robustness & diagnosis
2. Resource management
3. Programming
3. Organization & deployment

3.3 Nomadic together with Private / Home

The four top priorities, in decreasing order of priorities are :

1. Resource management
1. Organization & deployment
3. Global connectivity
4. Programming

3.4 Analysis

Resource management and programming are of top importance for all application domain clusters.

Provably correct systems priority is the highest priority for the critical application domain cluster, but does not rank as high for the other clusters.

Robustness & diagnosis have a very high importance for both Critical & Device & plant cluster.

Organization & deployment have a very high importance for both Device & plant cluster and the Nomadic and Private/home clusters.

4 DETAILING OF HIGHEST PRIORITIES

4.1 Resource management

As embedded systems become more seamlessly connected to each other, they are expected to be more and more subject to changes in their physical and logical environment. They are expected to dynamically adapt to such changes. Adapting their execution to the changing environment will be more efficient than applying too pessimistic hard real-time dimensioning techniques, but such dynamicity is a high challenge for real-time embedded systems.

Resource management is needed in such scenario for ensuring that the resource reserves or budgets are guaranteed. It will allow to achieve a high utilization of the system resources such as CPU, memory, network, and energy, in order to enhance the overall system performance. Also, it will distribute and allocate system resources according to the application requirements. For this purpose, resource usage accounting, budget enforcement, and monitoring are essential mechanisms to be provided by the real-time kernel.

Following topics were recognized as important issues to be solved:

- Self-organisation
- Self Calibration
- Very large scale dynamic network of embedded devices: environment awareness, ad-hoc networking
- Power & energy management techniques
- New OS's for SoC, more modular (fine-grain resource mgmnt) and service-oriented
- Mix of time-critical and best-effort implementations
- Dynamic allocation of applications onto multi-core platforms taking into account functional as well as non-functional aspects

4.2 Robustness & diagnosis

Robustness is the capability of a system to deliver an acceptable level of service despite the occurrence of transient or permanent hardware faults, design faults, imprecise specifications, and accidental operational faults. The research challenge is to devise middleware services that improve the robustness of the infrastructure services and support developers in ensuring robustness of application services.

In conjunction with architecture-level mechanisms (e.g., structuring of the overall system into fault containment regions that fail independently), middleware can contribute to achieving the goal of developing a robust system.

Following topics were recognized as important issues to be solved:

- Detection, Diagnosis, Error Containment and Fault Handling
- Safety-critical (cost challenge: avoid explosion of nr of boxes)

4.3 Programming

Requirement for the middleware is to provide application developers with a modular programming model that explicitly states inter-module dependencies, or in other words that formalise the "integration contract" of application software components. This enables development teams to work independently from each other while keeping the ability to seamlessly integrate the application, based on well-defined inter-module interactions. Modules are hereafter referred to as components.

The other key requirement is to enable application developers to focus on the functional, business behaviour of the application, while letting implementation of non-functional/technical aspects of the application up to the middleware. In other words, the middleware shall enforce a strong separation of concerns. Indeed, targeted applications have a wide variety of non-functional requirements. For instance, some applications may require the underlying middleware to support fault-tolerance and strongly encrypted communication (e.g. a card payment terminal), while others would require no security (e.g. an audio device commenting art pieces in a museum).

Due to versatility of underlying hardware and communication mechanisms, it is not possible for an application developer to foresee every single potential physical configuration the software may be deployed on. Middleware will have to come up with solutions making the fewest number of assumptions on the underlying hardware. The proper compromise between static middleware configuration and dynamic middleware adaptation undoubtedly depends on the application and the specific domain. As a consequence, middleware might be able to expose different domain-specific flavours. This is a necessary condition for making the middleware applicable to the different application domains targeted by ARTEMIS.

Additionally to these high-level requirements, one can identify several other important features the middleware might support. First, the inherent complexity of targeted application advocates for a hierarchical component composition mechanism that would enable applications, services, subsystems and systems to be composed in an abstract manner, regardless of the technicalities involved in managing the underlying platforms. Ability to consider several components as a single, coarser-grained component is essential to reasoning about assemblies and applying properties to those assemblies. If several middleware technologies already support such hierarchical composition mechanism, doing so without sacrificing memory footprint and real-time performances is still an open problem.

Moreover, the programming model of the middleware might support a wide variety of architectural styles, such as workflows, dataflows, semi-structured event-based interactions, interactive applications, stream-based multimedia applications, cooperative multi-user applications, and so on. Today's middleware usually focus on a small set of architectural styles, and this contributes to limiting integration capabilities.

To sum up, the general philosophy of research orientations presented above is to make middleware become the cornerstone of a declarative application development paradigm: the application components' internal and external structure, the non-functional features they require, as well as the middleware's own customisation and configuration shall be stated declaratively rather than programmatically. In conjunction with code generation techniques, and thanks to future enrichment of catalogues of available middleware services, the declarative approach to application development may dramatically contribute to increase productivity of software product vendors and software-based service providers.

Following topics were recognized as important issues to be solved:

- Architecture Description Language
- Implementation agnostic modularised elements
- New OS's for embedded systems, more modular (fine-grain resource mgmnt) and service-oriented
- Support of virtual machines
- Component based design

4.4 Organization & deployment

Pervasive computing infrastructures are by definition highly distributed and dynamic. In order to successfully realize applications for such a fluent environment, application developers need software technologies that are able to manage the adaptation, computation and communication requirements in an efficient and transparent manner. The requirements for such pervasive middleware technologies vary significantly across the different application domains.

In recent years, a substantial number of generic agent communication languages as well as different agent platforms that support interoperability of heterogeneous networked devices and applications have been proposed. On the other hand, specialized middleware concepts for the management of pervasive computing issues are being developed by a number of research groups. As the number of pervasive middleware concepts is constantly growing, it becomes increasingly important to develop a common understanding of the mandatory feature set and to identify suitable solution concepts.

Following topics were recognized as important issues to be solved:

- Construction and mgmnt of distributed computing objects
- Dynamic reconfigurable structures/applications
- Ontologies / semantic support
- Device and service discovery
- Conflict resolution
- Efficient user interaction / multi-modal

4.5 Provably correct systems

Building distribution platform for seamlessly connected systems is a complex task. One has to cope with the restrictions enforced to achieve (real-time) embedded systems, or to meet stringent requirements. Thus, one has to be able to assert middleware properties, e.g. functional behavioural properties such as *absence of deadlocks*, *request fairness*, or correct *resource dimensioning*; but also *temporal* properties, to validate real-time properties.

Following topics were recognized as important issues to be solved:

- Operating Systems & middleware for safety critical systems
- Designing and integrating provably correct systems
- Platform independent certification

4.6 Global connectivity

For the vision of seamless connectivity, we also need to identify, compose, configure, and maintain a multitude of interconnected embedded systems, each with different capabilities. These systems will have to locate and recognize objects and people and to analyze the context, adapt, and learn from the users around them. Today's, most embedded systems and devices are not aware of their environments and therefore cannot make timely, context-aware decisions. This is an architectural shortcoming of today's embedded systems. Intelligent environments are also prerequisites to meet the challenge of seamless connectivity.

Connectivity has to be enabled across borders between embedded systems & subsystems, networks, services and environments with seamless handover between heterogeneous access schemes and sessions. These technologies may include the current wired and wireless technologies by adding some extra functionality on higher layers of software and hardware implementations.

There is no single wired or wireless access or radio technology to provide system connectivity in all scenarios and in all application domains. In future, we will face heterogeneous networks that include some of the current wireless access schemes in addition to some advanced complementary technologies even for niche scenarios and application cases.

Following topics were recognized as important issues to be solved:

- Service-based architecture : "security & service platform"
- Connectivity in constrained environments

5 RECOMMENDED APPLICATION DRIVERS

The following list captures application drivers that participants to the ARTEMIS Summer Camp felt important to target, for development and valorisation of the ARTEMIS developed technologies.

Note: These are SUGGESTED, NON-PRIORITISED APPLCATIONS which can serve to use/prove results of ARTEMIS developed technologies. This is NOT an exclusive list, and it does NOT represent a mandatory set of requirements for research projects.

5.1 Transportation

Safety and efficiency

Also covering systems for engine/motor control for fuel efficiency and reliability

Travel assistance

5.2 Production and logistics

Covering the integration of autonomous process controls, sensor arrays, RFID etc…

5.3 Social and Health solutions

Mobile, remote, stationary

5.4 Communication Networks

Autonomy, self-organisation, ubiquitesness, security

5.5 Smart Systems

Sensors and Actuators
Evolution of Smart Cards