European Union
6th Framework Program

Information Society Technologies

Framework for Real-time Embedded Systems based on COnTracts

FP6/2005/IST/5-034026

Deliverable: D-RA.1

Requirement Analysis - Phase 1

Responsible: Gerhard Fohler

Authors: all FRESCOR partners

21 December 2006

Purpose & Scope

Deliverable on Requirements Analysis version 1

License and Distribution

Permission granted for receivers to re-distribute this document for use in the public domain except for the logos that belong to their respective owners.

Destination	Function/role	For Action	For Information
FRESCOR partners		X	X
European Commission		X	X

Revision history

File: **D-RA1**

Release/Status	Date	Author/Reviewer	Comments
1.0	2006-11-07	Gerhard F	consolidated input by all
1.0	2006-10-14	Gerhard F	First template

FRESCOR project (FP6/2005/IST/5-034026) is funded in part by the European Union's Sixth Framework Programme. This work reflects only the author's views; the EU is not liable for any use that may be made of the information contained herein.

Contents

1. Introduction 4

2. Wide Requirements Analysis Matrix 5

3. Requirements List 7

3.1. Requirements group 1 - Ability to compose components with their own timing and resource usage requirements 7
3.2. Requirements group 2 - Distribution (of components over standalone computers connected via network) 17
3.3. Requirements group 3 - Contract support 18
3.4. Requirements group 4 - Task model recurrent, periodic, sporadic, aperiodic, bursty 26
3.5. Requirements group 5 - Timing and synchronization constraints 30
3.6. Requirements group 6 - Support for tracing and profiling 37
3.7. Requirements group 7 - HW platforms 38
3.8. Requirements group 8 - RTOS 47
3.9. Requirements group 9 - SDK 51
3.10. Requirements group 10 - Energy management 55

4. Conclusion 56

1. Introduction

This deliverable presents the first version of the requirements for the FRESCOR project. It contains the results of a wide collection, analysis, and selection of application requirements for the project. The goal has been to wide collection of requirements from the involved and other industrial partners, both FIRST (IST-2001-34140) and OCERA (IST-2001-35102) EU IST projects, and a workshop held with selected participants from the ARTIST2 Network-of- Excellence. The focus of the deliverable is thus on collection, not on structure in this phase, which will be the focus of the second phase.

Position of deliverable in project

This deliverable forms the basis for the work in the first of two phase in the FRESCOR project and feeds directly into the other workpackages. The experiences gathered in this period will provide the basis for a revisiting of the requirements for the second phase. Thus all workpackage will provide input for the next version of this deliverable.

This overview is be too wide and unspecific. In order to set concrete and realistic goals for the project, a second step will analyse this wide collection, provide structure, and select specific requirements to be addressed in the project in the form of profiles.

Approach

The requirements have been collected from the aforementioned sources, a listing of input sources and text can be found in the appendix. For each requirement, the following properties were identified: input source, type, application area, priority, and related tasks. The resulting listing is represented in matrix form, called the *Requirements Analysis Matrix (RAM)*.

From RAM and input source text, a harmonized listing of requirements with common format, following the matrix grouping was produced.

Structure of the deliverable

The deliverable gives the Requirement Analysis Matrix first in section 2, followed by a detailed listing in section 3.

2. Wide Requirements Analysis Matrix

The Requirement Analysis Matrix (RAM) was created from listing requirements according to basic properties:

- *Input source*
  gives the source of the requirement to be a project or company.

- *Type*
  can be “temporal”, “platform” - relating to operating system and implementation issues, or “others”

- Application area

- *Priority*
  indicates the importance with respect to achieving the project objectives of the requirements – at this point, with likely modifications for the second phase. It can be “high” - essential for the project, “medium” - interesting but not necessary, “low” - to be done if time, “maybe” - to be decided.

FRESCOR Requirements Analysis Matrix **requirements on**

Number	Requirement	Input source	Type	Application area	Priority	Tasks
1	ability to compose components with their own timing and resource usage requirements	FIRST	temporal			
1.1.1	ability to support independently developed components/applications	FIRST	temporal		High	EP3, CF2
1.1.2	ability to support independently developed components/applications	FIRST	temporal		High	EP3, CF3
1.2	Offline composition	FIRST	temporal		High	AT1, AT2
1.3	online composition	FIRST	temporal		High	AT3, EP3
1.4	support for resource sharing between components	FIRST	temporal		High	EP3, EP6
1.5	single and multiple tasks within one component	FIRST	temporal		High	EP3
1.6	support for protection boundaries for synchronization and sharing of resources	FRESCOR	temporal		Medium	EP3
1.7	adaptation of system (resource allocation to components)	FRESCOR, OCERA	temporal	Multimedia, AI, control,...	High	EP3
1.8	isolation of resource usage of components	FIRST	temporal		High	EP2, EP3, EP4
1.9	Provide support for platform instantiation for contract	ARTIST	temporal		Medium	CF2, EP3, AQ1
1.10	Support dynamic software upgrade of components	Rapita	temporal	Telecom and space	Low	EP3
1.11	Standard compliant component model	OMG specification	platform	SDR	High	CF3
1.12	Standard compliant deployment	OMG specification	platform	SDR	High	CF3
1.13	Declarative application level contracts	FRESCOR	platform		High	CF2
1.14	Component timing analysis	FRESCOR	temporal		High	CF2
1.15	Component timing simulation	FRESCOR	temporal		High	CF2, OP2
1.16	Include analysis of overheads such as bus contention and others	FRESCOR	temporal		High	EP3
1.17	ability of components to know current resource availability	FIRST	temporal		High	
2	distribution (of components over standalone computers connected via network)	FIRST	platform			
2.1	Support for distributed negotiation	FRESCOR	platform		High	ND4
2.2	Support reconfigurable distributed architectures (node joining and leaving)	FRESCOR	platform		Low	ND3, ND4
3	contract support	FIRST	platform			
3.1.1	ability to do online acceptance tests	FIRST	temporal		High	AT1, AT2, AT3
3.1.2	ability to do offline acceptance tests	FIRST	temporal		High	AT1, AT2, AT3
3.2	support (in contract) for components with multiple levels of resource demands	FIRST	temporal		High	EP3, EP4, EP5, ND2
3.3	support for dynamic renegotiation	FIRST	temporal		High	EP3
3.4	support for bounded resource usage of negotiations	FRESCOR	temporal		High	EP3
3.5	Bounded response time for negotiation	Thales	temporal		High	EP3
3.6	support for controlling of stability of resource allocation	FRESCOR	temporal	distributed applications	Medium	EP3, ND2
3.7	The ability to specify the minimum duration of the contract	FRESCOR	Temporal (qos)	Mobile devices	Low	EP3
3.8	Specify memory as part of the contract (permanent and active)	FRESCOR	Others	Multimedia systems, mobile applic	High	EP4
3.9	Specify memory allocation rate	FRESCOR	Others	Multimedia systems, mobile applic	Medium	EP4
3.10	support for various system wide optimization goals	FRESCOR	temporal		High	EP3, AQ3
3.11	Support for multiple contract management	Thales	platform	SDR	Medium	EP3
4	task model recurrent, periodic, sporadic, aperiodic, bursty	FIRST	temporal			
4.1	Variable periodicity, continuous, discrete	FIRST	temporal		High	EP3, ND2
4.2	ability to express variable resource usage (continuous and discrete) per task invocation	FIRST	temporal		High	EP3, ND2, EP4. EP5
4.3	Handle data/time dependent resource usage	FIRST	temporal	Video processing	Medium	EP3
4.4	Ability to express relation btw power configuration and execution time (including CPU, bus, etc	FRESCOR	temporal		High	EP3, ND2, EP4, EP5
4.5	support for components with anytime, imprecise, and n-version algorithms	FIRST	temporal	AI, video processing	High	EP3, ND2
4.6.1	Transactions oriented architectures	FIRST	temporal		High	ND3
4.6.2	Data-flow oriented architectures	FIRST	temporal		High	ND3
4.7	Support for interrupt handlers (run-time support, budgeting and analysis)	Thales	platform		High	EP3
5	timing and synchronization constraints	FIRST	temporal			
5.1	Dynamic importance/weight provide means for differentiation between components (in relation t	FRESCOR	temporal		High	EP3
5.2	hard deadlines	FIRST	temporal		High	EP3, ND2
5.3	Detect and notify thread deadline misses	FIRST, OCERA	temporal		Low	EP3
5.4	Detect and notify „server" deadline misses	FIRST, OCERA	temporal		High	EP3
5.5	Supported application level Quality of Service (QoS) attributes	FIRST, OCERA	temporal		High	EP3, EP4, EP5, ND2, AQ3
5.6	end-to-end deadlines	FIRST	temporal		High	ND5
5.7	intermediate deadlines	FRESCOR	temporal		Medium	ND5
5.8	Precedence constraints	FIRST	temporal		High	EP3, ND2
5.9	Precedence between components	FRESCOR	temporal		High	EP3, ND2
5.10	mutual exclusion	FIRST	temporal		High	EP3
5.11	input/output jitter control	FRESCOR	temporal		High	EP3
5.12	Support for wide range of time granularity	Thales, OCERA	temporal		High	EP3
5.13	Support throughput requirements	Rapita	temporal	Telecom	High	EP3, ND2
5.14	The ability to specify requirements in the application domain (at least for multimedia)	FRESCOR	Temporal (qos)		High	AQ1, AQ2, AQ3
6	support for tracing and profiling	FRESCOR	platform		High	EP3
6.1	ability to attach additional profiling (and other) information to contract	FRESCOR, OCERA	platform		Low	EP3
6.2	support for hardware tracing tools	FRESCOR	platform		Low	EP3
6.3	On-line monitoring of resource usage (including detection of budget overruns)	FRESCOR, OCERA	platform		High	EP3
7	HW platforms					
7.1	Homogeneous multiprocessors – no migration	FRESCOR	platform		Maybe	EP3
7.2	Homogeneous multiprocessors – migration in case of mode changes	Magneti Marelli	platform	automotive	No	EP7
7.3	Heterogeneous multiprocessors – no migration	Evidence	platform		Maybe	EP7
7.4	USB 2.0 protocol and driver	FRESCOR	platform		High	EP3, ND2
7.5	Support for various CPUs: Pentium, ARM, MIPS, PPC	ALL	platform		High, High, L	EP3, EP4, EP5
7.6	wired ethernet	FRESCOR, OCERA	platform	Control	High	ND2
7.7	fieldbus	OCERA	platform	Control	Low	ND1
7.8	(wireless) 802.11	OCERA	platform		Low	ND3
7.9	TCP/IP	FRESCOR, OCERA	platform		High	ND2
7.10	support for disk bandwith management ATA, SATA	VT	platform	Video processing	Low	EP n
7.11	Bus bandwidth management (PCI)	VT	platform	Video processing	No	EP n
7.12	Recovery resources on abnormal termination or exception	Rapita	temporal		High	EP3
8	RTOS					
8.1	ENEA OSE	Enea				
8.2	OSEK/VDX	Magneti Marelli, Evidence	platform	automotive	Low	EP7
8.3	support for real-time execution of applications at user level	VT, OCERA	platform	Video processing	High	EP3
8.4	Support for heterogeneous application sizes (kb to Mb)	Thales	platform		High	EP3, EP4, EP5
8.5	Scalability of the framework to very large systems (hundreds or thousands of threads)	FRESCOR	platform	Telecom Base station	Medium	EP3, ND2
8.6	global time base, monoton clock	VT				
9	SDK	FRESCOR	Others			
9.1	GNU/Linux supported by Frescor Framework	VT, OCERA	Others		High	EP2, EP3, EP4, EP5, EP6
9.2	Component framework supported programming languages	FRESCOR, OCERA	platform	SDR	High	CF3
9.3	Evalutate the performance and quality in a simulated environment	Rapita	temporal		High	OP1-OP4
9.4	Design space exploration via simulation	Rapita	temporal		Medium	OP5
9.5	Integration with modelling tools (Matlab/Simulink)	Thales	Others		Low	OP6
10	Energy management	ARTIST	Energy			
10.1	System-wide power management	FRESCOR	Energy		High	EP5, AQ3
10.2	Support for controlling the power state for external devices	FRESCOR	Low		High	EP5
10.3	Support a discrete number of power states for the processor/memory/bus/network	FRESCOR	Low		Low	EP5

3. Requirements List

3.1. Requirements group 1 - Ability to compose components with their own timing and resource usage requirements

Requirement 1.1.1

Title: Ability to compose independently developed components/applications

Text: The FRESCOR framework shall provide means to compose applications and components on the same processing resource.

Motivation: FRESOR is aimed at supporting more than one component/application at a time

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, CF2

Priority: High

Requirement 1.1.2

Title: Ability to develop independent components/applications

Text: The FRESCOR framework contract shall allow each application to specify its own temporal and resource requirements.

Motivation: Applications can be independent of each other, therefore they have to be able to specify their own temporal and resource requirements.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, CF2

Priority: High

Requirement 1.2

Title: Offline composition

Text: The necessary tests and analysis for composition is undertaken before any application runs. There are no new components added to the original set at run-time.

Motivation: This the simplest form of composition and is appropriate for some application domains.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: AT1, AT2

Priority: High

Requirement 1.3

Title: Online composition

Text: As new applications/components are added to a running system the necessary admission tests are undertaken, and if passed the new entities are linked into the existing system.

Motivation: Necessary part of the framework to support dynamic composition

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: AT3, EP3

Priority: High

Requirement 1.4

Title: Support for resource sharing between components

Text: To be able to deal effectively with resources shared between applications

Motivation: It is unrealistic to assume all applications are totally independent of each other, some level of sharing is to be expected.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, EP6

Priority: High

Requirement 1.5

Title: Single and multiple tasks within one component

Text: The FRESCOR framework shall provide support for multithreaded applications.

Motivation: To restrict all applications to be sequential would have a serious impact on the expressive power of the framework

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3

Priority: High

Requirement 1.6

Title: Support for protection boundaries for synchronization and sharing of resources

Text: When an application composed of several components is running it is necessary that they have mechanisms to share resources in such a way that the resource is protected from concurrent accesses with the goal of keeping its overall consistency.

The protection mechanism for shared resources must be such that an erroneous or malicious component cannot exceed the usage of the resource by some allocated maximum.

Motivation: Shared resources are a common requirement in concurrent real-time software. Even in the case of independently developed components, it is necessary for them to cooperate and used shared resources. Shared resources need some kind of synchronization mechanism that guarantees the internal consistency of the resource. Two main mechanisms can be used for this purpose

- wait-free synchronization mechanism: it is usually based on keeping several copies of the shared resource, to allow concurrent accesses, each on a different copy; the drawback is that the overhead may be very high
- mutual exclusion synchronization: concurrent accesses are forbidden or controlled in such way to guarantee the data consistency. Overhead is usually small, but the technique introduces blocking times during which the resource may be unavailable for some components.

The wait-free mechanisms don't need any specific protection, but the more efficient mutual exclusion mechanisms do. The fact that a component may hold the lock on some resource for a long time may cause other components to fail. Therefore, a mechanism is needed to specify budgets for the usage of the resource (measured in execution-time), enforce these budgets, and take some corrective action if they are exceeded, consisting of rolling the state of the shared resource back to a safe value.

This protection mechanism is unnecessary for those components that have passed a static analysis that guarantees that the shared object will be used according to the established budget. This may be the case for high-integrity components. For them it should not be necessary to pay the price of setting and enforcing budgets on shared resource usage.

Input source: [From York, Assorted thoughts about requirements for FRESCOR]

Shared objects: either no mutual exclusion or budgets needed and roll-back mechanism.

[From FRESCOR proposal]

An integrated view of all the resources is required: in addition to the processor and network resources, there are many resources that should become part of the set with real-time scheduling requirements, to achieve full independence of the components used to build an application. This integrated view should include all system resources in addition to the threads and networks: dynamically reconfigurable modules, interrupts with time protection, shared resources with time protection, memory protection, and energy/power-aware scheduling.

Type: Temporal

Application area: All

Related Tasks: EP3

Priority: Medium

Requirement 1.7

Title: Adaptation of system (resource allocation to components)

Text: It is necessary that while an application is running there is some mechanism in place to modify and renegotiate the contracts made for the different application components to adapt their characteristics according to the actual resource consumption, with the goal of optimizing the application's quality of service.

Motivation: The resource requirements on many applications, such as multimedia, are usually not known a priori, and may greatly vary during runtime depending on external factors. Consequently, the initial resource allocation made in the contracts negotiated with the system may not be appropriate to achieve the desired quality of service. By monitoring the actual resource usage the contracts may be adapted to the current system load with the goal of optimizing the quality of service.

Input source: [From York, Flexibility Requirements from the AI Domain], OCERA

Advanced task execution monitoring is needed for more precise control of resources for supporting value-based scheduling. Valuable information about resource allocation at the system level should be collected and shared by the OS in a consistent manner. More resource-aware AI applications can then alter its task parameters through the negotiation interface depending on the status of resource usage in the system. With these features a feedback-control mechanism can be constructed where dynamic, adaptive applications can best make use of a flexible resource management framework.

[From FRESCOR project proposal]

A QoS manager is required as a middleware layer that "mediates" between the application and the underlying operating systems. In particular, the QoS manager must translate the high level QoS requirements of the application into resource allocations. As an example of high level QoS requirements for a video/player or audio/video communication system, we must consider minimum frame/rate and video resolution, sound quality, decoding quality, multimedia content. Applications such as video streaming are expected to be run on a wide range of hardware platforms and reconfigure themselves depending on the available resources. It is rarely known a priori, during system development, on which hardware the application will be run. Moreover, the resource requirements of these applications depend on lot of external factors that may greatly vary during run-time.

Using classical real-time theory on this application is simply not possible mainly for time-to-market and cost reasons. In particular, estimating the worst-case resource requirements (i.e. execution time on a processor) is a difficult and time-consuming activity. Moreover, resource requirements may vary during run-time making it impossible to analyze off-line all possible configurations. Finally, such application allow a certain degree of constraints violation (i.e., deadline misses) as far as the overall QoS of the application is respected.

It is necessary that the QoS manager is able to dynamically adapt the resource allocations in order to optimize resource utilization (e.g., to optimize energy) without sacrificing on QoS requirements.

Type: Temporal

Application area: Multimedia, AI, control,...

Related Tasks: EP3

Priority: High

Requirement 1.8

Title: Isolation of resource usage of components

Text: Temporal encapsulation of subsystems is required to support the composability of independently developed components. This encapsulation is achieved by isolating the components from each other in their usage of resources such as processor capacity, network bandwidth, shared resource usage, memory, and energy.

Motivation: The isolation of components is essential in any component-based methodology, to achieve the goal of being able to deploy components developed by independent parties. It is necessary that a malfunctioning component does not steal resources from other well-behaved components.

Input source: [From FIRST Deliverable D.AF1v2 "Final Architecture Framework Definition"]

3.5.1 Robustness and Isolation. The operating system will prevent applications from suffering due to other applications over-running for example. A system of budgets may be used to enforce this.

[From FRESCOR project proposal]

Temporal encapsulation of subsystems is required to support the composability of independently developed components.

Proper abstractions are required. The design methodology should take care of transforming the high-level timing requirements into the low-level RTOS primitives in a transparent way. It should be able to measure and estimate timing properties with little intervention from the developer, and apply automatic real time analysis, as well as provide performance and quality of service estimations. It should provide fault detection and protection from faults in other components with high-level constructs. Its middleware layers should not hide away those implementation and hardware-specific details that are essential to have the ability to reason about the timing behavior.

Type: Temporal

Application area: All

Related Tasks: EP2, EP3, EP4

Priority: High

Requirement 1.9

Title: Provide support for platform instantiation for contract

Text: When designing and implementing a component, or part of the application, the programmer does not yet know which platform the component will be run upon. Therefore, it is not possible to completely specify the resource requirements of the contract at design and implementation time. Our FRESCOR framework has to provide means to only partially specify a contract. In particular, the resource requirements of the component (like for example the minimum and maximum allocated budgets) need to be instantiated later, possibly at deployment or even at run-time.

Motivation: One of the objectives of the proposed framework is to provide support for component-based design and development of real-time systems. A component will be designed and developed in isolation and then re-used in different applications and possibly on different hardware platforms. To

be able to control the temporal requirements of the component, we will associate one or more contracts to a component. Such contracts are part of the component specification and implementation. However, the designer and the programmer does not yet know the resource requirements of the component. In particular, if the component is re-used on a different platform, its resource requirements are likely to change.

Input source: [ARTIST workshop on Requirements]

Type: Temporal

Application area: All

Related Tasks: CF2, EP3, AQ1

Priority: Medium

Requirement 1.10

Title: Support dynamic software upgrade of components

Text: The framework will allow real-time software components to be upgraded at run-time, without violating timing requirements.

Motivation: Software components can be upgraded "patched" on during run-time, for example to correct errors or to provide additional functionality. The implications for timing analysis are that the patched software should continue to perform correctly, including any specified timing behaviour after the patch. There are some hidden problems here, for example, replacing a routine by another one can increase the execution time of other (functionally independent) parts of the system, because the new software may interfere in a variety of ways, including using different cache lines or changing bus loading. Such problems can potentially be identified before the patch is released, but the ability to reconfigure the timing requirements of all parts of the system is a run-time issue directly relevant to FRESCOR.

The motivation for this requirement comes from the telecommunications industry and the space industry. Concrete examples are mobile telephones (which can receive new software updates and bug fixes from the network automatically), mobile telephone base stations (which can be upgraded to support new tariffs, routing, network structure information and bug fixes on the fly from a remote server without engineers visiting the base station sites or shutting down services), and satellites (which can receive software upgrades to support new functionality and bug-fixes) after they have left the earth and physical interaction with the device is impossible).

Requirement 1.11

Title: Standard compliant component model

Text: The component framework developed as part of WP3 shall be conformed to the OMG lightweight CCM specification.

Motivation: This is a strategical choice in order to have some visibility, work in the scope of a community, and commercially supported implementations incorporating the extensions to be proposed by the project.

Input source: OMG specification

Type: **P**latform

Application area: SDR

Related Tasks: CF3

Priority: High

Requirement 1.12

Title: Standard compliant deployment

Text: The application deployments shall follow the OMG Deployment and Configuration specification (D&C)

Motivation: This is a strategical choice in order to have some visibility, work in the scope of a community, and commercially supported implementations incorporating the extensions to be proposed by the project. Besides, some add-on can be done, due to the new requirements coming from FRESCOR (for example: on-line deployment and (re-)configuration)

Input source: OMG specification

Type: Platform

Application area: SDR – multimedia applications etc.

Related Tasks: CF3

Priority: High

Requirement 1.13

Title: Declarative application level contracts

Text: The project shall provide an abstraction to annotate component based application description by a set of scheduling contracts. Referring to requirement1.9, a specification for subsets of contracts linked to application modes could be provided.

Motivation: Component based applications needing to leverage the flexible scheduling framework, should come together with specification of contracts, instead of relying directly to « frsh » API defined in the project to inform the negotiation and allocation engine about its requirements. The idea is to have direct interaction with the « frsh » API encapsulated in some component runtime module interpreting the scheduling contracts specification and performing the negotiation.

Input source: FRESCOR

Type: Platform

Application area: All

Related Tasks: CF2

Priority: High

Requirement 1.14

Title: Component timing analysis

Text: Automation of timing analysis of software components given a set of constraints shall be provided.

Motivation: In order to cover the whole real-time software engineering aspects, it is needed to perform timing analysis (e.g. WCET analysis like those referred in req. 5.4). This task can be simplified thanks to component code generation, that could provide instrumentation associated to components, and an environment to analyze their timing isolation. More precisely in the scope of the project, a tool (integrated or leaved external) has to be used to realize off-line time analysis. This one can be, either a commercial one (like *Rapitime* or VirtualTime) or, a specific tool provided by FRESCOR project. For instance, it can be based on a trace system, and/or some results coming from WP9-AT tasks that could be integrated in the framework.

Input source: From FRESCOR project proposal

Type: Temporal

Application area: All

Related Tasks: CF2, AT1, AT2, AT3

Priority: High

Requirement 1.15

Title: Component timing simulation

Text: Integration of timed simulation capabilities in the component framework shall be considered.

Motivation: This is a very value added tool that can be reliable and accurate to ease real-time software integration. The idea is to integrate individual temporal behavior for components (obtained as detailed in Requirement 1.14 above) automatically in a simulation of a component based application. More precisely in the scope of the project, a tool (integrated or leaved external) has to be used to realize on-line analysis. This one can be, either a commercial one (like VirtualTime) or, a

specific tool provided by FRESCOR project. It based for instance, on an enhanced trace system, or results coming from AT, OP tasks that could be integrated in the framework.

Input source: From FRESCOR project proposal

Type: Temporal

Application area: All

Related Tasks: CF2, OP2, AT1, AT2, AT3

Priority: High

Requirement 1.16

Title: Include analysis of overheads such as bus contention and others

Text: The timing model used in the framework must take into account that certain operations, such as bus arbitration or memory copies are not instantaneous and their timing must be included in the overall analysis.

Motivation: Video processing applications must move substantial amount of data (e.g. in memory-to-memory copies). In other real-time applications (e.g. real-time control) one can neglect these operations because the time required is small enough. However, one must consider these operations in the analysis of video processing applications.

Input source: FRESCOR

Type: High

Application area:

Related Tasks: EP3, AT1-n

Priority: High

Requirement 1.17

Title: Ability of components to know current resource availability

Text: There shall be a mechanism for application components to obtain the current level of resource availability, to adapt their own execution requirements to the available capacity

Motivation: When an application component has the ability to choose between different operational modes to adapt itself to the available resources, it is necessary that a run-time mechanism exists for the component to determine the current level of available resources and select an operational model in accordance to it.

Input source: FIRST project requirements

Type: Temporal

Application area: All

Related tasks:

Priority: High

3.2. Requirements group 2 - Distribution (of components over standalone computers connected via network)

Requirement 2.1

Title: Support for distributed negotiation

Text: It is necessary that distributed systems are supported in the context of the scheduling framework. This means that it is necessary that the network (or networks) is one of the managed resources for which the usage from each component is guaranteed and controlled. It is also necessary that the resource usage for a distributed application can be negotiated in a global way, because there are implicit dependencies between the components distributed across different processing nodes.

Motivation: Many real-time applications are distributed, and therefore support for distributed applications is a general requirement for a scheduling framework.

Input source: [From the presentation by Paulo Pedreiras, University of Aveiro]

Modern embedded systems show a higher tendency to become distributed systems. Safety concerns have led to mostly static approaches, but there is a growing interest in using dynamic operational scenarios, to accommodate for changes and efficiently use resources.

There is the need to dynamically and flexibly manage the network bandwidth to support periodic and non periodic traffic, with variability in the periodic behaviour possible through some controlled stability state.

[From FRESCOR project proposal]

At the scheduling domain, it is necessary to manage the contracts of a distributed transaction, in distributed reconfigurable architectures. A general truly distributed manager is required to dynamically manage transactions on a global system basis. The distributed transaction manager must be capable of addressing the distributed consensus problem of negotiating contracts in a distributed environment.

Type: Platform

Application area: All

Related Tasks: ND4

Priority: High

Requirement 2.2

Title: Support reconfigurable distributed architectures (node joining and leaving)

Text: The scheduling framework must support the possibility of nodes in a distributed system leaving the system, or joining it. Other nodes must be capable of knowing the network configuration at run-time.

Motivation: Wireless networks are a very dynamic distributed environment in which it is very likely that mobile element will leave or join the system, either permanently or on a temporary basis (for instance because of a temporary loss of signal). In wired networks it is also possible to find dynamic scenarios in which a variable number of clients may join the system.

Input source: [From the presentation by Paulo Pedreiras, University of Aveiro]

Modern embedded systems show a higher tendency to become distributed systems. Safety concerns have led to mostly static approaches, but there is a growing interest in using dynamic operational scenarios, to accommodate for changes and efficiently use resources.

There is the need to dynamically and flexibly manage the network bandwidth to support periodic and non periodic traffic, with variability in the periodic behaviour possible through some controlled stability state.

[From FRESCOR project proposal]

Reconfigurable architectures are a promising technology for embedded systems because they provide the adaptivity and dynamicity that a complex application with Quality of Service requirements demands.

The distributed nature of the application makes it necessary to take into account distributed reconfiguration strategies, dynamically adapting the system timing requirements to the available processors and networks. This is especially important in wireless environments where nodes may dynamically enter or leave the system.

Type: Platform

Application area: All

Related Tasks: ND3, ND4

Priority: Low

3.3. Requirements group 3 - Contract support

Requirement 3.1.1 (UY)

Title: Ability to do online acceptance tests

Text: The FRESCOR framework shall perform online acceptance tests on the submitted tasks during run-time, if online the acceptance test is enabled.

Motivation: Fundamental to the contract model is the notion of an acceptance test that must be passed before applications can be admitted. Only if the new and old timing requirements can be met will the new entity be accepted. Note, acceptance may change the quality of service of existing applications.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: AT1, AT2, AT3

Priority: High

Requirement 3.1.2

Title: Ability to do offline acceptance tests

Text: The FRESCOR framework shall provide support for static acceptance test.

Motivation: Due to potential high processing demand of the online acceptance test, there is also a need to support offline acceptance tests.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: AT1, AT2, AT3

Priority: High

Requirement 3.2

Title: Support (in contract) for components with multiple levels of resource demand

Text: The FRESCOR framework contract shall provide the ability for applications to specify multiple system resource requirements.

Motivation: As part of the contract, an application can specify a range of resource needs. The implication is that the more resources the run-time can provide the better, but the minimum amount is adequate. Considerable flexibility comes from being able to specify different levels of service and for run-time decisions to be made about the level to support.

Review sheet note: The possible co-domain (continuum or a set of discrete levels) of resource requirements is specified in requirement 5.2.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, EP4, EP5, ND2

Priority: High

Requirement 3.3

Title: Support for dynamic renegotiation

Text: An application should be able to renegotiate its contract at run-time.

Motivation: As applications run and learn more about their execution profile they may benefit from a new contract. Applications may also incorporate mode changes that will require renegotiation. Of course the renegotiation may not be successful.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3

Priority: High

Requirement 3.4

Title: Support for bounded resource usage of negotiations

Text: The overhead that the negotiation process imposes is bounded in all the resources (nodes, networks, memory, energy,...) and this bound is controllable as a configuration parameter.

Motivation: The need for bounded negotiation overhead comes from the general need of being able to guarantee the timing requirements of the application. All elements running in the system must have bounded resource usage to achieve this general requirement, and the contract negotiation is no exception.

Moreover, the related requirement 3.5 requires the ability to have some bound on the contract negotiation time, and this implicitly requires some way of controlling the amount of resources that can be allocated to the negotiation process.

Input source: [From the presentation by Luis Nogueira, Polytechnic Institute of Porto]

In distributed systems, there are requirements for handling heterogeneous nodes, where each node may supply a distinct set of resources, with different capacities.

The system is dynamic, with tasks arriving and leaving while others are being executed. Each service has user-specific QoS constraints, all competing for a finite set of resources. Negotiation of services is a cooperative task. We usually need to find the subset of nodes that performs the service with the best quality, allowing for some degree of quality tradeoffs, between some bounds.

A good solution, not necessarily optimal is enough, and lets us bound the amount of resources spent in negotiation. Anytime algorithms are best for the management of the framework.

Type: Temporal

Application area: All

Related Tasks: EP3

Priority: Medium

Requirement 3.5

Title: Bounded response time for negotiation

Text: The response time required to perform a negotiation, or a renegotiation, is bounded. This includes the time required for the new resource allocation resulting from the negotiation process to enter into effect.

Motivation: Many applications require rapid changes to their resource allocation (for instance due to some mode change) or rapid admission of a new component. The time required to perform a negotiation must be bounded and this bound must be known to those applications.

Moreover, the bound must be controllable through some mechanism, usually by changing the allocation of resources for the negotiation process itself (see requirement 3.4).

Input source: [From Thales requirements for software defined radio]

Requirement TCF-SDR-12. Predictable reconfiguration time

Upper bounds formulas on the contracts negotiation times shall be obtained. In a radio system, in the scope of reconfiguration of the communication link, all communicating actors have to agree on a «timing protocol» to reconfigure partially, if nothing specific (like resynchronization, ...) is planned in the waveform.

Type: Temporal

Application area: All

Related Tasks: EP3

Priority: High

Requirement 3.6

Title: Support for controlling of stability of resource allocation

Text: The mechanism to control the allocation of resources to the negotiation process must be stable in a such way so that small change in the system does not lead to unbounded reallocation of resources.

Motivation: Resource allocation may be seen as an optimization/decision problem which respects nature of tasks to be performed, character of available resources and optimization/decision criterion. Such problems, usually specified by the set of constraints are usually solved by some iterative algorithms with proven convergence, Artificial Intelligence methods, Mathematical programming, approximation algorithms or various heuristic techniques. Convergence and performance of these algorithms are two main issues guaranteeing stability of the resource allocation process.

Existing scheduling theory covers relatively well the case of centralized optimization/decision problems where the algorithm is run at one place (i.e. all algorithm input and output data are available at that place, and the algorithm is executed on one machine). The situation is different in the case of our general problem, where some of the data, resources, and constraints have local sense (e.g. CPU) while the others do represent an intersection among several nodes or even have a global sense (e.g. network). Generally speaking we are faced to distributed optimization/decision problem that can be solved in several ways. One way is to collect all the data in one node, to solve the problem by centralized algorithm, and to distribute the solution to all nodes. The second way is to make all the data available at all nodes, to run the algorithm on each of them and to iterate the negotiations in order to solve the conflicts. The third way is to solve the problem in a distributed manner where only relevant data are available at a given node, to solve the "local" problem with respect to a given constraints, to propagate such solution in the form of additional constraints to other nodes, to augment the set of constraints and to repeat this process.

Input source: FRESCOR

Type: Temporal

Application area: Distributed Applications

Related Tasks: EP3,ND2

Priority: Medium

Requirement 3.7

Title: The ability to specify the minimum duration of the contract

Text: The application shall be capable of specifying the minimum duration that a contract should be in place, and the system should accept this contract only if it can guarantee that with the current

energy resources it can fulfill that application requirement.

Motivation: The total energy available in a system running on batteries is usually known to some degree of precision. An application may want to run some process only if it knows that it can run it for some interval of time that is enough to achieve its objectives. This should then become a parameter to the contract. The system can accept this contract if it knows that with the available energy and the current policy for optimizing energy it can guarantee the requested minimum time. It will also reject other future negotiations that would require a change to the energy consumption policy that would violate the already accepted contracts.

Input source: Discussions at the requirements workshops on requirements derived from the ability of the system to optimize the usage of the available energy.

Type: Temporal (QoS)

Application area: Mobile devices

Related Tasks: EP3

Priority: Low

Requirement 3.8

Title: Specify memory as part of the contract (permanent and active)

Text: An application will require for its execution an amount of memory to allocate all needed resources. Dynamic memory management is used by the application tasks to allocate pieces of memory (application objects) that will be released in the future. An application component has to specify in the contract the maximum amount of memory needed at each job invocation it needs to allocate. The system has to grant these resources to each negotiated contract.

Application components can allocate memory in two different ways: (i) permanent and (ii) temporal or active. A permanent memory allocation means that the component is owner of this allocated memory and can release it at a specified time. When a component has exhausted its specified needs, it can, additionally, use more memory in a temporal way that requires to be granted by a quality of service-server related to dynamic memory management. This active memory has a temporal use determined.

The contract model should be able to differentiate between algorithm-related memory which may change, and application execution-related memory such as stacks or special variables, which cannot be changed in a complied version of the code. A contract may have to be rejected if e.g., stack sizes are incompatible.

Motivation: Any approach to real-time memory management must be able to ensure that the application does not run out of memory and has sufficient usage of the CPU. The specification of the memory required by an application is very relevant in embedded applications. In order to use properly the system resources, the application has to determine which are the dynamic memory needs for its execution. Dynamic memory plays an important role in modern software engineering paradigms (object oriented paradigm) and techniques (Java technologies). Moreover, it allows to

increase the flexibility and functionalities of the applications.

However, in the real-time community the use of dynamic memory techniques has not been considered due to some misunderstandings about dynamic memory management. With the existence of allocators that performs the allocation/deallocation in constant time, the use of dynamic memory in real-time systems is an important issue.

The flexibility granted by the system on other resources as CPU, network, etc, can be extended to the memory management.

Input source: [From the presentation by UPVLC, Workshop on Requirements held in Paris in conjunction with FRESCOR meeting]

Discussions with industrial partners in the Workshop on Requirements

Type: Basic contract parameters

Application area: Multimedia systems, mobile applications, soft real-time applications

Related Tasks: EP4

Priority: High

Requirement 3.9

Title: Specify memory allocation rate

Text: An application will require for its execution an amount of memory to allocate all needed resources. Dynamic memory management is used by the application tasks to allocate pieces of memory (application objects) that will be released in the future. An application component has to specify, in the contract, the maximum amount of memory needed. The system has to grant these resources to each negotiated contract.

Application components can allocate memory in two different ways: (i) permanent and (ii) temporal or active. A permanent memory allocation means that the component is owner of this allocated memory and can release it at a specified time. When a component has exhausted its specified needs, it can, additionally, use more memory in a temporal way that requires to be granted by a quality of service server related to dynamic memory management. This active memory has a temporal use determined.

Motivation: In a multimedia scenario, several video sources can be processed to be shown in the monitor or to be recorded or even analyzed to perform some actions. Each task involved in the image acquisition allocates pieces of memory for each video frame and stores this frames in a buffer. Another task gets frames form the buffer and process them. Both task involved in the processing, with the buffer, can be considered as one application that requires allocation of memory. In order to analyze the system needs, a limit on the rate of allocation has to be provided. This allocation rate can be specified in bytes per second or bytes per period.

Input source: [From the presentation by UPVLC, Workshop on Requirements held in Paris in conjunction with FRESCOR meeting].

Discussions with industrial partners in the Workshop on Requirements.

From the use of dynamic memory management in the Realtime Java specification.

Type: Basic contract parameters

Application area: Multimedia systems, mobile applications, soft real-time applications

Related Tasks: EP4

Priority: Medium

Requirement 3.10

Title: Support for various system wide optimization goals

Text: The system shall support the run-time specification of various system-wide optimization goals including at least optimizing the power consumption, the level of resource usage, or combinations among them.

Motivation: Using the information available in the contracts that have been negotiated in the system, it is possible to distribute the assignment of system resources in various ways. The particular choice should depend on the system-wide optimization goal specified by the system manager through some API. In the FIRST project there was only one system-wide optimization goal which was maximizing the resource usage. The spare capacity, and possibly dynamically reclaimed capacity, was made available to the application components if they had expressed their ability to make use of this extra capacity in their contracts. In FRESCOR we should support additional system-wide optimization goals, such as reducing the power level to the minimum, or some combination: not exceeding some power level and trying to maximize the resource usage up to that level.

Input source: ARTIST-2 workshop on flexible requirements, and FRESCOR proposal.

Type:

Application area: All

Related tasks:

Priority: High

Requirement 3.11

Title: Support for multiple contract management

Text: An application could be executed in different modes, each associated with sets of tasks / contracts. The latter shall be provided, both programmatically at scheduling contract framework level, and declaratively at the component framework level.

Motivation: Radio applications can have multiple modes, for which different kinds of data are

carried. Examples are: idle, synchronization seek, (data or voice) reception, (data or voice) emission, full-duplex operation, and so forth.

Input source: Thales – Real-time embedded applications

Type: Platform

Application area: SDR

Related Tasks: EP3

Priority: Medium

3.4. Requirements group 4 - Task model recurrent, periodic, sporadic, aperiodic, bursty

Requirement 4.1

Title: Variable periodicity

Text: The framework should provide support for tasks with variable periods; the definition of variable task periods either as a continuum between two limits (the elastic model) or as a set of discrete values.

Motivation: Applications define their requirement via a task (or job) model. A common need is to support periodic tasks. Variable periods can be defined as either a continuum between two limits (the elastic model) or as a set of discrete values. The implication is that the more frequent the execution the better, but that the slowest rate is adequate.

Power-train control applications in automotive require to control the timing of the spark in the cylinder. Such applications usually include a task that is activated every half rotation of the crank-shaft. Therefore, the activation rate of this task is time-variable and directly proportional to the current rate-per-minute (RPM) of the engine. To manage overload situations due to a high RPM, it is important to provide contract support for this kind of tasks.

Input source: Many application areas including AI, Automotive applications

Type: Temporal

Application area: including AI, Automotive (power train)

Related Tasks: CF1, EP3

Priority: Maybe

Requirement 4.2

Title: Ability of express variable resource usage (continuous and discrete) per task invocation

Text: In addition to expressing a range of requirements on the overall contract, an application may express a specific need on a per task (job) execution. The per task request must be compatible with the contract previously negotiated.

Motivation: Increased quality of service can be delivered if a fine level (per task) control over resource usage can be undertaken.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, EP4, EP5, ND2

Priority: High

Requirement 4.3

Title: Handle data/time dependent resource usage

Text: A task/job may consider its data or the current time and then request a budget based on an estimate of its needs. This is a special case of 5.2

Motivation: Although traditional real-time systems have a constant resource need per task, the framework here will support more flexible and dynamic requirements.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: Video processing

Related Tasks: EP3

Priority: Medium

Requirement 4.4

Title: Ability to express relation between power configuration and execution time (including CPU, bus, etc)

Text: It shall be possible to specify in the contract the relationship between the resource usage times (for instance execution times in the processors) and the power level. Since this relationship is complex, it may be given at several discrete power levels at which particular values have been measured.

Motivation: Power management requires the control of mechanisms such as CPU clock that have an impact in the overall processing power available (MIPS) impacting thereby the execution time of all the tasks. The implication is that worst-case execution times are a function of the power management state at any given time. Therefore, the framework must be able to express the execution time as a function of the power management configuration.

Input source: FRESCOR

Type: **Temporal**

Application area: Video Processing, all

Related Tasks: EP3, ND2, EP4, EP5

Priority: High

Requirement 4.5

Title: **Support for components with imprecise, anytime and n-version algorithms**

Text: The FRESCOR framework shall provide support for components with AI algorithms.

Motivation: The ability to effectively exploit spare capacity at run-time. A number of algorithms exist that enable an application to benefit from extra computing resources been made available at run-time. Three broad classes of these algorithms are imprecise, anytime and n-version algorithms.

Input source: Analysis undertaken as part of the FIRST project, and a further study of the AI area.

Type: Temporal

Application area: AI and Video processing

Related Tasks: EP3, ND2

Priority: High

Requirement 4.6.1

Title: **Transactions oriented architectures**

Text: The FRESCOR framework shall support transaction oriented architectures

Motivation: Applications do not just consist of independent tasks but chaining relationships such as transactions.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: ND3

Priority: High

Requirement 4.6.2

Title: Data flow **oriented architectures**

Text: The FRESCOR framework shall support data flow oriented architectures

Motivation: Data may also be defined to flow through a number of components to form a dataflow oriented architecture.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: ND3

Priority: High

Requirement 4.7

Title: Support for interrupt handlers (run-time support, budgeting and analysis)

Text: Interaction of tasks (taken in charge by contracts) with interrupt service routines or more generally event driven application shall be considered.

Motivation: Typical radio transceivers are hardware interrupt based, and adequate sequencing for signal chunks transmission interrupts versus tasks to feed transceiver is needed.

Input source: Thales - SDR applications

Type: Platform

Application area: SDR, as well as all applications running on hardware communicating with FPGA and DSP chipsets

Related Tasks: EP3

Priority: High

3.5. Requirements group 5 - Timing and synchronization constraints

Requirement 5.1

Title: Dynamic importance/weight provide means for differentiation between components (in relation to any resource)

Text: The FRESCOR framework shall allocate spare capacity of any resource to the application based on the applications attributes "importance" and "weight/quality".

Motivation: Many applications can take advantage of spare capacity to improve their computation results.

Requirement 5.2

Title: Hard Deadlines

Text: The framework must be able to support applications that have hard deadlines.

Motivation: Although there is a focus on flexibility, hard deadlines are still important and must be supported

Input source: Analysis undertaken as part of the FIRST project

Type:: Temporal

Application area: All application areas.

Related Tasks: EP3, ND2

Priority: High

Requirement 5.3

Title: Detect and notify thread deadline misses

Text: The framework must provide a way to detect and notify when an activity such as a computation or a message sent through a network, misses its deadline.

Motivation: When developing any real-time application, it is important to be able to detect and timing anomalies, both during testing and debugging, and at run-time. In the second case, by detecting a failure of a thread to miss its deadline, it could be possible to take some appropriate action to prevent any system failure.

Detecting deadline misses is also important for soft real-time systems. In the case of a high number of deadline misses, the component could decide to re-negotiate the contract to achieve a higher QoS.

Input source: FIRST, OCERA.

Type: Temporal

Application area: All

Related Tasks: EP3

Priority: Low

Requirement 5.4

Title: Detect and notify "server" deadline misses

Text: To be able to detect and notify to the programmer the fact that a "server", i.e., the run-time implementation of a contract misses its deadline. A server that misses a deadline is a failure of the framework, so it is quite important to be able to detect such a failure.

Motivation: A server may miss its deadline when the framework is not correctly implemented, or when some wrong assumption is made on the application. In particular, if two applications/components, that are served by different servers share some resources, it may happen that one application holds the critical section for more than declared. If the mechanisms described in Requirement 1.6. "Support for protection boundaries for synchronization and sharing of resources" is not implemented, the temporal protection is broken and some server can miss its deadline.

It is very important to be able to detect this situation as it is a break of the temporal isolation properties, and invalidates the framework assumptions.

Input source: FIRST, OCERA

Type: Temporal

Application area: All

Related Tasks: EP3

Priority: High

Requirement 5.5

Title: Supported application level Quality of Service (QoS) attributes

Text: The FRESCOR framework shall provide support for the following application level QoS requirements:

1) Control application related

    (a) absolute release jitter

(b) absolute finishing jitter

(c) end-to-end delay

(d) sample period

2) Multimedia processing and transmission related

(a) resolution

(b) frame-rate

(c) video and audio synchronization

(d) bandwidth (average value and peak rate)

(e) maximum burst size (token bucket parameters)bit-rate (video bandwidth expressed in Kbits / second)

(f) end-to-end delay (maximum, average and jitter)

(g) output rate (this value can be expressed in Kbits / second)

(h) packet loss rate

3) Scheduling related

(a) soft deadline

(b) firm deadline

(c) n in m task completion constraint

Motivation: Various forms of quality of service support should be provided. On flexible systems the utilization of various resources can vary as long as the user perceived quality does not fall below a lower bound.

Input source: Analysis undertaken as part of the FIRST, OCERA projects, applications.

Type: Temporal

Application area: Multimedia- and control applications

Related Tasks: EP3, EP4, EP5, ND2, AQ3

Priority: High

Requirement 5.6

Title: End-to-end deadlines

Text: As well as activity-specific deadlines there will be deadlines spanning a sequence of activities that must be supported.

Motivation: Many applications require guarantees not only for single tasks or messages, but over a sequence of tasks and messages. While the completion time of tasks within the sequence may vary, the time between start of the first activity and completion of the last has to meet a certain deadline. Typically, this is used for the time between input to output of transactions.

Examples for end-to-end deadlines come from control, where the time between start of sensing and completion of actuation is relevant, not the completion times of the individual tasks; or media processing, where the transmission of a stream is bound by the display time on a terminal device, rather than individual tasks.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: ND5

Priority: High

Requirement 5.7

Title: Intermediate deadlines

Text: When end-to-end deadlines of sequences of activities are given, intermediate deadlines, which relate to the start of the overall sequence have to be kept.

Motivation: In addition to individual deadlines of tasks, and end-to-end deadlines referring to sequences of activities, intermediate deadlines relate to the start of the overall sequence. They can stem from, e.g., allocation of tasks to nodes, where a sequence of activities spans over several nodes, with an end-to-end deadline. Then, the completion of tasks on a node can be limited by intermediate deadlines for nodes.

Intermediate deadlines are also a means to decompose end-to-end deadlines in distributed systems.

Input source: FRESCOR

Type: Temporal

Application area: All application areas

Related Tasks: ND5

Priority: Medium

Requirement 5.8

Title: Precedence constraints

Text: The FRESCOR framework shall provide support for precedence constraint among tasks.

Motivation: There is a need to support precedence constraints since certain tasks depend on their predecessors. The dependent tasks must not start until all their predecessors have completed their execution. This is a common requirement in applications and usually associated with end-to-end deadlines, which are defined over the sequence of precedence constrained tasks.

One question relates to the periodicity of tasks with precedence constraints. When the periods of these tasks are the same, precedence is clear. When the periods defer, the semantics needs further exploration.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3, ND2

Priority: High

Requirement 5.9

Title: Precedence between components

Text: Components that require input from the results of execution of other concurrent components need some way of wait synchronization that allows this data flow requirement to be fulfilled.

Motivation: The need to exchange data among components, and in general to be able to support transactions with some data flow architecture introduces the need to support precedence relations among components, which can be achieved via a wait synchronization mechanism.

Input source:

[From FRESCOR project proposal]

It is necessary to provide for a transparent management of the quality of service at the system's level, by providing an end-to-end distributed transaction manager, adaptive system-level quality of service management middleware, and system-level design and optimization methods.

Type: Temporal

Application area: All

Related Tasks: EP3, ND2

Priority: High

Requirement 5.10

Title: Mutual exclusion

Text: Need to support the synchronization constraint of mutual exclusion.

Motivation: A common requirement in applications.

Input source: Analysis undertaken as part of the FIRST project

Type: Temporal

Application area: All application areas

Related Tasks: EP3

Priority: High

Requirement 5.11

Title: Input/Output jitter control

Text: The FRESCOR framework shall allow each application to specify the permitted range of task release jitter.

Motivation: Control applications usually can tolerate task release jitter only within a specified range. Both, too early and too late release of certain tasks can lead to unusable results.

Requirement 5.12

Title: Support for wide range of time granularity

Text: The framework shall support variable time granularity in order to allow the adjustment of system clock precision on different targets. As a matter of fact, this (low level) feature can be provided in another way: the framework could support a wide range of periods and deadlines. At execution time, a given CPU clock precision could be fixed, or, if we consider req. 5.4, this CPU clock precision could be changed depending on the power management state.

Motivation: This feature will permit to support multiple kinds of application constraints.

Input source: Thales Real-time embedded applications, previous meetings, OCERA

Type: temporal

Application area: All

Related Tasks: EP3

Priority: High

Requirement 5.13

Title: Support throughput requirements

Text: The framework will support the definition of throughput-based timing requirements; a throughput-based requirement considers a number of (usually sequential) events, for example: "n requests per second".

Motivation: In telecommunication base stations, with many parallel threads handling simultaneous connections, there are both deadlines on the individual jobs, and more general throughput requirements. Particularly for systems with many parallel threads, QoS requirements can be expressed as throughput conditions on overall behaviour, rather than (for example) deadlines on individual threads. This allows a more system-level description and provides some flexibility. As an example, consider a server handling some soft real-time requests; a natural description of the timing requirement is "must handle at least 100 requests per second". Perhaps with some understanding of the distribution of the incoming requests, a throughput-based description allows the system to exploit flexibility in scheduling, allowing some jobs to take longer than others, rather than a stricter "each job just take less then 1/100 second".

Input source: Rapita

Type: Temporal

Application area: Telecommunications

Related Tasks: EP3 ND2

Priority: High

Requirement 5.14

Title: The ability to specify requirements in the application domain (at least for multimedia)

Text: The definition of QoS parameters could be given as framework parameters or as application domain parameters.

Motivation: From the point of view of the developer, it is easier and more powerful to specify QoS levels as application domain parameters. Sometimes is difficult to match domain requirements to framework values. It is difficult to adapt parameters when we change between different supported platforms (x86, XScale, ...).

Input source: FRESCOR

Type: Temporal(qos)

Application area:

Related Tasks: AQ1, AQ2, AQ3

Priority: High

3.6. Requirements group 6 - Support for tracing and profiling

Requirement 6.1

Title: Ability to attach additional profiling (and other) information to contract

Text: A contract involve the definition of different parameters related to a concrete service. This parameters can be defined by the application or can be provided through a profile information obtained from the system execution.

Motivation: The definition of a contract can be complex for a set of applications that use the system resources in different ways. For this kind of applications the behaviour could be described better by a system resources measurement obtained in previous (test phase) execution.

Input source: FRESCOR, OCERA

Type: Platform

Application area: General purpose

Related Tasks: EP3

Priority: Low

Requirement 6.2

Title: Support for hardware tracing tools

Text: Trace allows developers to collect information about the state of the CPU over a period of time. The system will support specific hardware tracing tools to monitor the system status.

Motivation: In the past, hardware logic analyzers collected CPU state information for analysis. Hardware logic analyzers became limited because they relied on the ability to monitor external address and data busses for information. As processors became more complicated, more functionality was performed internally, and logic analyzers were unable to adapt since signals and state information previously available to logic analyzers became hidden. Modern trace hardware, supported by many processors has made it possible to gain access to these internal signals without disturbing the system or surrendering correctness. New trace collection and analysis techniques use this trace data to reconstruct the state of the processor and provide in-depth visualization of system behavior.

Input source: FRESCOR

Type: Platform

Application area: General purpose

Related Tasks: EP3

Priority: Low

Requirement 6.3

Title: On-line monitoring of resource usage (including detection of budget overruns)

Text: The ability of a system to monitor the resource used by applications is one of the most attractive and needed features in a flexible system. Main system components as: CPU, Memory Network, Energy, etc. or other aspects as: response time measurement, system load, budget overrun, etc., are needed.

Motivation: Tracing facilities allows the user to automatically trace the real-time application (without any manual instrumentation) in order to compute interesting metrics such as task computation times, resource usage, blocking factors, etc. The final result of the metrics processing should be an easy-to-read report including the analysis of the application's temporal behaviour. Tracing facilities are based on a light version of the POSIX Tracing services and have to be integrated in the kernel. With these services, execution profiles of the application and on-line measurement can be used.

Input source: FRESCOR, OCERA

Type: Platform

Application area: General purpose

Related Tasks: EP3

Priority: High

3.7. Requirements group 7 - HW platforms

Requirement 7.1

Title: Homogeneous multiprocessors – no migration

Text: This requirement deals with simple communications between same kinds of processors. In many use-cases there are no needs to transfer tasks from one processor to another.

Motivation: In SDR use-cases, there could be some General Purpose Processors connected together. The application just need to communicate in a distributed context.

Input source: from FRESCOR project proposal

Type: Platform

Application area: All

Related Tasks: EP3

Priority: **Maybe**

Requirement 7.2

Title: **Homogeneous multiprocessors – migration in case of mode changes**

Text: In homogeneous multiprocessor systems (e.g. SMP), it may be necessary to migrate a task from one processor to another one in case of a structural load change, as it may happen during a mode change. The framework must be able to support such migration efficiently. In particular, the contract that the task has established with the system must be valid also after migration.

Motivation: In the next future, some automotive supplier is forecasting the use of SMP technology for increasing the power of the HW platform without increasing the processor frequency. In such a case, some of the added features of a SMP platform are the increased flexibility of re-organizing the task allocation at run-time. Power-train applications present a wide range of load conditions, depending on the particular mode of the system and on the engine RPM. Therefore, it would be useful to do some load balancing on the two processors in case of a structural load change when changing mode.

The framework should be able to support such load-balancing facility without compromising the basic contract guarantees.

Input source: Magneti Marelli – Power Train

Type: Platform

Application area: Automotive

Related Tasks: EP7

Priority: No

Requirement 7.3

Title: **Heterogeneous multiprocessors – no migration**

Text: There is the need to support applications running on heterogeneous multiprocessors systems, i.e. Systems consisting on one micro-controller plus one or more co-processors (DSP or similar) dedicated to specific activities.

Motivation: Many embedded hardware platforms consist of one micro-controller plus one or more dedicated processor (DSP or similar). The system may be treated as a distributed system-on-a-chip. However, compared to classical distributed systems, heterogeneous multiprocessors may

communicated by shared memory. In such a case, it is possible to optimize communication by using shared memory primitives rather than using higher-level communication mechanisms. Our framework should be able to support such a kind of system. In particular, by extending the contract framework to take into account heterogeneous systems.

Input source: Evidence

Type: Platform

Application area: All

Related Tasks: EP7

Priority: Maybe

Requirement 7.4

Title: USB 2.0 protocol and driver

Text: The driver and protocol of this kind of communication port shall be provided by the framework. USB is now available on some boards (e.g. PowerQuicc), it permits to exchange data between platforms with high rate.

Motivation: In SDR use-cases, providing that such a feature is available, it permits to simulate high speed links (like those in the UMTS stack).

Input source: from FRESCOR project proposal

Type: Platform

Application area: All

Related Tasks: EP3 - ND2

Priority: High

Requirement 7.5

Title: Support for various CPUs: Pentium, ARM, MIPS, PPC

Text: The framework should provide support for at least the most widely used hardware architectures and processors in the embedded systems world.

Motivation: Embedded applications run on a wide variety of processors and hardware platforms. Our framework must address this diversity by supporting at least the most widely used processors and architectures. While the Intel architectures are not widely used in embedded systems, it is easier to develop, test and debug the framework on top of Intel-based architectures (like normal PC). ARM

Ltd. provides a popular family of processors for embedded systems, as well as Motorola PowerPCs and MIPS.

Input source: All

Type: Platform

Application area: All

Related Tasks: EP3, EP4, EP5

Priority: High for Pentium and ARM, Low for MIPS and PPC

Requirement 7.6

Title: Wired Ethernet

Text: The framework will work on real-time distributed systems, different networks and protocols will be supported including widely used Ethernet.

Motivation: A cheap and well supported equipment, high and steadily increasing transmission rates, a simple binding with protocols on higher levels of network stack, easy to install and understand are arguments why wired Ethernet became very popular and therefore it needs to be supported as a primary network medium.

To support network contracts an underlying communication layer must meet the following requirements (not only for Ethernet, can be applied to fieldbus and wireless networks too):

- **hot-plugging**
  - nodes join/leave the network dynamically
- **support for different types of traffic**
  - real-time (synchronous, asynchronous) , non-real-time
- **traffic isolation**
  - various types of traffic do not disturb each other
- **support for upper-layers of the framework**
  - providing services to network reservation mechanism, spare network capacity distribution, Qos management, flexibility, on-line adaptability
- **support for multicast messages**
  - has to be considered since it may be required by some high-level layers
  - **fault-tolerance**

Input source: FRESCOR, OCERA

Type: Platform

Application area: Control

Related Tasks: ND2

Priority: High

Requirement 7.7

Title: Field bus

Text: Field bus support is required for interoperability of existing specially designed hardware. Field buses follow industrial environment requirements and they are able to connect computer based systems with real word sensing and actuation. Therefore field bus is one of the network supported by FRESCOR.

Motivation: Many of the embedded control systems are distributed today and very important part of these systems is the field area.

The fieldbus link layers and protocols have been designed with heavy duty, response time, interference elimination and durability in mind. Even that many conventional fieldbuses are replaced by Ethernet for its lower infrastructure costs and higher transmission rates, there are many situations still, which require to connect into legacy infrastructure or where Ethernet does not offer advantage or is even problematic to use. The Ethernet communications and associated protocols complexity requires extensive amount of memory in small sensors or actuators for example.

Our long time experience with CAN and Profibus technologies enables us to consider more tight integration of these solutions with ongoing project. As the first considered task, Linux CAN driver frames prioritization and infrastructure enhancements could provide base for better achieving timing control and sufficient bandwidth allocation control.

Due to their deterministic nature (contrary to Ethernet) and existing response time analysis, the fieldbus systems can be easily integrated within FRESCOR contract based mechanism.

Input source: OCERA

Type: Platform

Application area: Control

Related Tasks: ND1

Priority: Low

Requirement 7.8

Title: wireless communications

Text: Wireless networks gained a favour with the following advantages: easy installation, mobility, flexibility, scalability, reachability, environment isolation (e.g. safe/unsafe part in gas refinery, clean/unclean environment in hospital, automotive e-safety systems, etc.). They are becoming so important in embedded systems today. The framework will support network contracts on wireless networks in the similar way like Ethernet, in addition it needs to support specific routing protocols of sensor networks.

Motivation: Wireless networks allow users to access information and services regardless of their geographical position and pre-deployment infrastructure. More important parameters for these types of networks over wired networks are varying time delay, jitter, packet loss and bandwidth. These requirements open new fields and approaches for decentralized and distributed control concept, autonomous strategies and real-time communication in wireless networks.

Two major wireless standards that operate in the 2.4 GHz ISM band are Wi-Fi and ZigBee. ZigBee is designed to monitor and control application satisfying the needs for Home Control, Building and Industrial Automation, while Wi-Fi networks are mostly locally organized for high data rates.ZigBee is a PAN wireless networking solution based on the IEEE 802.15.4 specification that supports low data rates, security, low complexity, easy deployment and self-organizing and self-healing large networks. The main advantages of ZigBee are the very low power consumption and the small footprint of the protocol stack, which make it possible to provide components to very low cost. ZigBee characteristics match with the requirements of wireless sensor networks.

Wi-Fi is a WLAN networking solution based on the IEEE 802.11 specification. The specification includes the RTS/CTS (Request to Send/Clear to Send) function to control station access to the medium. By the proper use of RTS/CTS, the operation of wireless LAN can be fine-tuned to solve the problem of hidden node.

IEEE 802.11e as of late 2005 has been approved as a standard that defines a set of Quality of Service enhancements for LAN applications, in particular the 802.11 WiFi standard. The standard is considered of critical importance for delay-sensitive applications, such as Voice over Wireless IP and Streaming Multimedia. The protocol enhances the IEEE 802.11 Media Access Control (MAC) layer.

Input source: OCERA

Type: Platform

Application area: Multimedia data transfers (video, voice), Home Control, Building and Industrial Automation, Automotive e-safety systems

Related Tasks: ND3

Priority: Low

Requirement 7.9

Title: TCP/IP family based communications

Text: Due to interoperability reasons, FRESCOR contract mechanism needs to support TCP/IP protocol family since it is widely used and well supported. The technology available at the market is mature and advanced. The framework must support this protocol family and it must provide the ability to send/receive message in real-time context.

Motivation: TCP protocol represents reliable transport service that is not suited for real-time messages transport because it favors reliability over timeliness. For example, when the data message containing measured temperature from sensor gets lost, a receiver asks the sensor to resend the same data even though the temperature may have changed meanwhile and the most recent value is available now. Also, while transmitting multimedia data (video, voice) it is not critical if some messages (within admissible rate) are not received. Retransmission of lost messages would break a desired fluent stream of data.

Consequently, the protocols RTPS (Real-time Publish-Subscribe), or its open implementation – ORTE , and RVSP(Resource reservation protocol) have been developed. RVSP gives a support to RTPS when data are transmitted through routers by reserving transmission capacity in routers along the data path. The Real-Time Publish-Subscribe communications model (RTPS) has proven to be a suitable model for real-time communications in Industrial Automation and other Distributed Control applications (e.g. Command and Control Systems, Telecom Network management). RTPS also specifies the way the application interacts with the middleware, that is, the types of communication entities and the properties and quality of service (QoS) parameters the application can request for each entity.

1 Gbps Ethernet , which provides large throughput with low latencies, with QoS mechanism could ensure soft-real-time data transfer even though TCP is used. The configuration and parameters for control devices need to be transferred reliable.

Input source: FRESCOR, OCERA

Type: Platform

Application area:

Related Tasks: ND2

Priority: High

Requirement 7.10

Title: Support for disk bandwidth management ATA, SATA

Text: Shape disk bandwidth between different applications.

Motivation: Disk bandwidth is one of the core resources of digital video recorders. Bandwidth shaping will guarantee enough bandwidth to store and retrieve video at the specified rates.

Input source: VT

Type: Platform

Application area: Video processing

Related Tasks: **EPn**

Priority: Low

Requirement 7.11

Title: Bus bandwidth management (PCI)

Text: A typical digital video recorder has several video boards plugged in the PCI system bus. The PCI bus is a scarce resources that must be carefully distributed among the different bus masters to ensure an optimal use. In particular, the ability to integrate the bus arbitration mechanism in the overall schedulability analysis would allow a cost-optimized implementation of digital video systems.

Motivation: Without the possibility to guarantee PCI bandwidth per video device it's possible to lose frames. PCI bandwidth control would guarantee requested fps.

It may be difficult to achieve, due to current PCI hardware does not usually provide mechanisms to control the bus arbitration from the application.

Input source: VT

Type: Platform

Application area: Video processing

Related Tasks: **EPn**

Priority: No

Requirement 7.12

Title: **Recovery of resources on abnormal termination or exception**

Text: FRESCOR will provide facilities to handle graceful return of resources on termination of an application including abnormal termination. Special handling may be required if the application fails during execution of a shared global resource.

Motivation: An application that terminates (whether abnormally or normally) should be able to release its resources (in terms of negotiated in contracts) back to the system for use in other applications. For applications that terminate normally, API calls to explicitly cancel contracts are quite practical, but for applications that fail and terminate in an uncontrolled way, the application cannot be relied upon to cancel it's contracts. Therefore upon the abnormal termination of a contract, a trusted entity outside of the application (perhaps the operating system) must be responsible for maintaining the contracts.

Input source: Rapita

Type: Temporal

Application area: All

Related Tasks:

Priority: High

3.8. Requirements group 8 - RTOS

Requirement 8.1

Title: ENEA OSE

Text: Part of the framework should be implemented on top of ENEA OSE Real-Time Operating Systems.

Input source: ENEA

Type: Platform

Application area: ENEA

Related Tasks: EP7

Priority: high

Requirement 8.2

Title: OSEK/VDX

Text: Part of the framework should be implemented on top of OSEK/VDX-compliant Real-Time Operating Systems.

Motivation: OSEK/VDX is a standard interface specification for Automotive applications. This interface is for completely static systems, where applications are completely configured off-line. However, such system have little flexibility, and no support for resource management, temporal protection and overload handling. While implementing the complete FRESCOR framework on a OSEK/VDX system seems inappropriate, a selected subset of functionalities would greately enhance the standard without adding too much overhead.

Input source: Magneti Marelli, Evidence

Type: Platform

Application area: Automotive

Related Tasks: EP7

Priority: Low

Requirement 8.3

Title: Support for real-time execution of applications at user level

Text: Add FRESCOR Framework support to tasks running at application level

Motivation: Common applications run at *application level*, different from *kernel level*. It will be useful to allow developers to access FRESCOR framework from application level in order to implement some soft real-time tasks.

The intention is that we support the contract model not only at the RT-level in RT-Linux, but also at the linux level. This is important for soft real-time applications that can tolerate the glitches caused by the unpredictability in the general Linux kernel (perhaps with some patches that help in reducing these uncertainties) becase in this way the application can make use of all the resources available to a general-purpose application, such as more networking facilities, graphical interfaces, file system, or energy-management primitives.

Input source: VT, OCERA

Type: Platform

Application area: Video processing

Related Tasks: EP3

Priority: medium

Requirement 8.4

Title: Support for heterogeneous application sizes (kb to Mb)

Text: Regarding supporting the execution of applications in a wide range of sizes.

Motivation: User applications can involve different needs in terms of code and data sizes. The operating system has to be able to execute application programs from Kbytes to Mbytes.

Input source: General requirements

Type: Platform

Application area: General purpose

Related Tasks: EP3, EP4, EP5

Priority: High

Requirement 8.5

Title: Scalability of the framework to very large systems (hundreds or thousands of threads)

Text: The framework will be scalable to very large systems with hundreds or thousand of threads.

Motivation: As an example, real model, when considering scalability of the framework to very large systems we can take large telecommunication systems. They require scalable and durable operating system for more different levels applications. The accurate billing system is one of them. Thousands of Call Detail Records (CDR) has to be collected in parallel with one millisecond resolution. The steady stream of records is collected into huge databases with reasonably small latencies. The other demanding tasks are connection target lookup operations which require fast database lookups as well. The other challenging tasks are multichannel audio or multimedia streams format conversions which allow to interconnect different technologies (VoIP, ISDN, etc). The usual today practice is to use specialized hardware for each of these services and tasks. The building of single image base system capable to run these tasks is demanding task. We can expect, that such system would utilize special DSP or FPGA based hardware accelerators to speed up some of the tasks.

We can consider next requirements for such base system:

1. The system has to be reliable and ensure, that there would not be starvation in ongoing communications. The contract based approach for resources allocations seems to be as a good principle to ensure that resources are not overloaded by means of precise resources counting and allocation. The number of parallel ongoing operations can be limited during exact task contract negotiation without unnecessary artificial limit caused by considering most demanding task (data conversion etc.) for each connection. The precise information about required resources could be used even for selection of different encoding techniques adjusted according to system load or remaining resources. The contract based ideas could even be used for accelerators and link bandwidth allocation.

2. The timing requirements of different tasks in such system could differ significantly. Some of the tasks require precise timing with short processing time (CDR). Other tasks require some bandwidth without strict release time requirements but with guaranteed data chunks processing completion times dictated by chosen data streams delays. The low bandwidth system management tasks and remote maintenance does not require any fast real-time guarantees in many cases, but it has to receive some reasonable amount of processor time because these tasks are extremely important to resolve some critical conditions and then they are more important than actual data streams. This leads to contradictions for classical priority based scheduler, because these task should receive high priority but in the fact they should run in the background. CBS, EDF or contract based CPU time distribution/scheduling could help to overcome these problems.

3. To ensure scalability of such system, the increase of algorithms complexity with increasing number of parallel connections has to be taken seriously. The ideal solution is to utilize algorithms with O(1) complexity in such system (for CPU scheduler, memory allocation, accelerators and datalink bandwidth allocation/sharing, etc.).

4. The use of runtime reconfigurable hardware accelerators could be considered as well. The accelerators could be based on reconfigurable FPGA design or they can be based on some

multicore design. The optimal division of data streams load between many (even differently or on demand programmed) units distributed in large system could be another challenging task for computer science theoreticians.

[1] Research on Real-Time Linux in Telecom Operating Supporting Systems

Ning Ting, Zhang Rui-sheng, Fan Xiao-liang, Liu Long-guang, Wang Hai-long

in Eight Real-Time Linux Workshop

Input source: FRESCOR

Type: Platform

Application area: Telecom Base station

Related Tasks: EP3, ND2

Priority: Medium

Requirement 8.6

Title: Global time base and monotonic clocks

Text: Ability to have a global time base in all nodes of a distributed multimedia system with monotonic clocks

Motivation: Multimedia control systems, in particular video surveillance systems, need to establish a correlation between video information stored by digital video recorders and the information acquired by other devices (e.g. text devices such as the Point-Of-Sales unit of a shop or a Programmable-Logic-Controller of a control installation). This information is required to make it easier the retrieval of video images corresponding to certain events generated by other devices. Typical examples are: video images corresponding to a certain transaction in a Point-Of-Sales unit (e.g. when a customer paid with credit card), or video images corresponding to an escalator stopping for an anomaly.

The most effective way to correlate video information with information from other devices is to design systems with a global time base (e.g. using NTP). Most of the applications analyzed require time synchronization in the range of 100ms (e.g. video traffic systems) to 1s (e.g. retail).

Digital video recorders used in video surveillance applications are programmed to record continuously the video images of the cameras connected to them. The ability to provide monotonic clocks avoids special *corner cases* to show up when the clock must be put back.

The ability to correlate video information with information from other sources will make it easier the data fusion required by future *cognitive systems* to interpret events in a human-like way.

Input source: FRESCOR

Type: Operating System support

Application area: Multimedia Processing

Related Tasks:

Priority: High

3.9. Requirements group 9 - SDK

Requirement 9.1

Title: GNU/Linux supported by FRESCOR Framework

Text: The FRESCOR framework running on to of linux platform.

The target kernel will be 2.6.X because almost all the new development are based on it.

We should take care of two scenarios, the hard real-time one and the soft real-time one. First will be based on RT/Linux. Some real time tasks, due to its timing and resources requested, should run at RT level because at application level is not possible to guarantee required resources and timing. In this case we will need FRESCOR framework on top of RT/Linux.

Soft real-time tasks tasks has lower timing and resources requirements. Because of that is possible to run these tasks at application level. In this case we will run FRESCOR framework on top of Linux (non RT).

Motivation: Linux is an open source operating system with several advantages: free access to the whole source code, a huge community developing drivers and application, completely customizable. Linux is one the best option for embedded devices.

Regarding kernel version, Linux 2.6 is the last kernel version. It is more robust than previous versions and it contains important features which could make it easier the implementation of the framework. The porting of Linux to some embedded processors, such as ARM-XScale, require the use of version 2.6.

Input source: VT, OCERA

Type: Others

Application area:

Related Tasks: EP2, EP3, EP4, EP5, EP6

Priority: High

Requirement 9.2

Title: Component framework supported programming languages

Text: C and C++ components implementation shall be supported

Motivation: They are the most commonly used languages in embedded and real-time software today.

Input source: FRESCOR, OCERA

Type: platform

Application area: SDR

Related Tasks: CF3

Priority: High

Requirement 9.3

Title: Evaluate the performance and quality in a simulated environment

Text: The timing behaviour of the framework will be able to be simulated using a host-based simulator product. This includes simulation of scheduling and applications, and the analysis of simulation results.

Motivation: Implementation of scheduling algorithms and applications on real hardware can be expensive and error prone. Further, there are tremendous difficulties to monitor the hardware in order to produce accurate, quantitative evaluation of the timing behaviour of the system. Simulation can provide accurate measurements and has lower development coasts. Therefore a simulation based-approach is advocated for prototyping and testing scheduling algorithms and applications, accurately comparing different approaches for evaluation purposes.

Input source: Rapita

Type: Temporal

Application area: All

Related Tasks:

Priority: High

Requirement 9.4

Title: Design space exploration via simulation

Text: The framework will facilitate the use of a simulation architecture to explore aspects of the design space, for example adjustment of scheduling parameters for optimal performance.

Motivation: For some applications, particularly where the timing requirements are difficult to define, for example applications described using QoS metrics, it is difficult to determine the best way to map the application to the underlying scheduling framework to give good or optimal performance. Simulation can provide an opportunity to explore how different software architectures or different mappings to the scheduling framework, or different scheduling parameters affect the timing behaviour and QoS of the application. As described in 9.3, scheduling provides a low-cost implementation route and facilities for accurate measurement. Simulation can be used repeatedly and the results of simulations compared easily. Therefore we advocate the use of simulation in a partially- or fully-automated exploration of appropriate areas of the design space.

Input source: Rapita

Type: Temporal

Application area: All

Related Tasks: 12.3

Priority: Medium

Requirement 9.5

Title: Integration with modeling tools (Matlab/Simulink)

Text: Matlab/Simulink provides a rapid prototyping and development environment, widely used for advanced control tasks.

Motivation: Matlab Simulink, allows engineers to develop a control application algorithm in high level graphical language of data-flow and state-flow diagrams. Tools like Real Time Workshop Embedded Coder and Stateflow Coder can be then used to automatically generate C code of the algorithm. However, the hardware and OS dependent parts of the application is not supported well. Only few microcontrollers are supported, portability is limited since blocks representing peripherals are different for different microcontrollers; they do not usually allow to precisely tuning all hardware parameters and no verification of this error prone process is done.

However, the implementation work is not only simple translation of the algorithm specified in details by the simulation model to the target language (usually C). Also many decisions related to non-functional aspects of the application as the hardware/software deployment, the scheduling policy, the fault tolerance mechanism etc. must be done. These decisions affect the behavior of the control algorithm understandably. Therefore the behavior of the implemented controller usually differs from the simulation model. The reason is that the real-time execution of the controller tasks on the microcontroller is accompanied by latencies and jitters. Also the precision of the microcontroller arithmetic is lower than the arithmetic of the simulation tool (if it is not simulated in bit-exact form) due to the lack of floating point arithmetic unit and shorter word width. Moreover, the test with the real plant in the loop usually shows that the implemented controller parameters must be returned

since they were tuned on the approximate model.

The quality of the generated code is comparable to the hand-written code. The generated code execution time is between 90% and 120% of the hand-written code and it requires approximately the same memory space as the handwritten code. The generated code is readable since it is commented by names of the model bocks and also names of variables are preserved. Contrary to the classical approach, the automatic code generation shortens the development time and reduces possible error sources, which leads to the development cost-reduction. The rapid application development approach does not bring only automatic code generation. It is a model based development method supported by a tool chain covering entire "V" model development chain. The tool chain consists of tools best suited for the specific development phase. Matlab Simulink has become standard for specification and modeling at different abstraction levels, control algorithm design and simulation. The code from the model can be generated either by Matlab Real-Time Workshop Embedded Coder, dSpace TragetLink or ETAS ASCET. The validation of each development phase is done by simulation in the Matlab Simulink. First "Model in the Loop" validate model of the controller. After code generation "Processor in the Loop" can be used to validate behavior of the real-time execution of the controller on microcontroller hardware in the loop with plant model in Simulink. Than "Hardware in the Loop" can be used to validate behavior of the entire control unit hardware and designer assumptions about peripherals circuit behavior. Finally the control unit function is validated in the real plant. All these phases are supported by Simulink and corresponding code generator target. The results of each experiment are used to continuous improvement of the Simulink model that remains still actual.

In the code generation process, the Real-Time Workshop (RTW) tool generates a C code for rapid prototyping. For highly optimized production quality code generation however, the add-on RTW Embedded Coder must be used. The tool StateFlow Coder is used for code generation from StateFlow charts. Except these tools, the platform dependent target is needed. The platform means a specific MCU with a compiler and operating system with a development tool. The target, except other, defines the language (C/C++), details about the MCU (8/16/32bit, little/big endian ....), and it automatically calls the development tools such as compiler and linker during the code generation process. The targets for RTW and RTW Embedded Coder differ.

There are several weaknesses of the existing approach. Only few targets exist and therefore far from all MCU families and derivates are supported. Each MCU target has its own block set. This fact prevents reusability and portability of the model using these hardware specific blocks. The way in which the peripheral hardware is handled by the generated code is predefined by the target developers and it can not be changed by the user – integration of application layers of industrial communication protocols would partially solve this problem. Validation of the hardware and OS settings in time and resource domain is missing. Each parameter change can therefore lead to an error prone process.

The introduction of some additional callbacks, marking support and informations could be used to extrapolate computation times and other contract like parameters of generated tasks bodies and could allow to automatically check design feasibility for given target platform.

There should be at least quickly looked at other more open and better modifiable options to Matlab/Simulink as are SCILAB/SCICOS for example.

The integration with blocks representing drivers and providing communication with real world should be taken into account. The Comedi provides access to many input/output cards. Some more tight integration and simplification of CAN driver and communication protocol use would help with fast applications design as well.

Input source:

Type: Others

Application area: All

Related Tasks: OP6

Priority: Low

3.10. Requirements group 10 - Energy management

Requirement 10.1

Title: System-wide power management

Text: System-wide power management, considering all power consumer and their interaction, as opposed to management of individual parts or devices, e.g., CPU.

Motivation: Many algorithms aiming at energy management focus on single parts of the system only, e.g., on the CPU. While these are obviously important aspects, they do no suffice alone. Rather, a system-wide approach is needed to include the energy overheads of algorithms, power consuming parts, such as memory or bus, as well the interdependencies of algorithms and devices with respect to energy.

Input source: FRESCOR

Type: High

Application area:

Related Tasks: EP5, AQ3

Priority: High

Requirement 10.2

Title: Support for controlling the power state for external devices

Text: The system should be able to control the energy of itself, but of external devices as well.

Motivation: The computing system may not be the only device drawing energy from the same source. Other devices, e.g., cameras, may be connected to the same battery. In such cases, energy management has to consider all energy consumers drawing from the same source.

Input source: FRESCOR

Type: low

Application area:

Related Tasks: EP5

Priority: High

Requirement 10.3

Title: Support a discrete number of power states for the processor/memory/bus/network

Text: The system has to consider discrete numbers of power states.

Motivation: Many power consumers cannot be run at unlimited numbers of power states with continuous range, but operate on relatively small numbers of states only. Energy management, in particular for system wide aspects has to take this into account for realistic algorithms. In addition, the dependencies of the resource usage by an application component on the currrnet power level may be difficult to define; a discrete number of power states helps in creating an association between power and resource usage, for instance by measuring at each of the defined power states.

Input source: FRESCOR

Type: Low

Application area:

Related Tasks: EP5

Priority: Low

---

4. Conclusion

This deliverable presented the results from the wide collection of requirements. These were grouped accordingly to type, application, and priority in the form of the Requirement Analysis Matrix. A detailed listing of requirements in common format was given.

This document forms the basis for the work in the project in the first phase and will be revisited with the experiences gained for the second phase.