EUROPEAN COMMISSION

Thematic Priority:
SIXTH FRAMEWORK PROGRAM

Priority 2.5.3
INFORMATION SOCIETY TECHNOLOGIES
Unit G3 Embedded Systems

Project Acronym:

SOCRADES

Project Full Title:

Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices

Proposal/Contract No: EU FP6 IST-5-034116 IP SOCRADES

Milestone M1

Specification of the service-oriented infrastructure

Status:	**Final**
Dissemination Level:	**CONFIDENTIAL**
Date:	**28.09.2007**

Organization Name of the Lead Contractor for this Milestone: **Tampere University of Technology**

Status Description:

Scheduled completion date[1]:	31.082007	Actual completion date[2]:	28.09.2007
Short document description:	The paper provides description of reaching the Project Milestone M1. The main responsibility for the generation the MS report was allocated to WP2.		
Author(s) deliverable:	Aleksandra Dvoryanchikova (TUT), Rajasekaran Andiappan (TUT), Oscar Perez (TUT), Bernard Bony (SE)	**Report/deliverable classification:** ☐ Deliverable ☒ Milestone Report ☐ Three-Monthly Activity Report ☐ Six-Monthly Activity Report	
☐☐ Partner ↓ Contributions ↓ Peer reviews	☒☐ Schneider Electric ☒☐ ABB ☒☐ APS GmbH ☐☐ Boliden AB ☐☐ FlexLink Automation Oy. ☒☐ Institut f. Automation und Kommunikation e.V. Magdeburg ☐☐ Kungliga Tekniska Högskolan	☒☐ Loughborough University ☒☐ Luleå University of Technology ☒☐ Politecnico di Milano ☒☐ SAP AG ☒☐ Siemens AG ☒☐ Tampere University of Technology ☐☐ Jaguar Cars Ltd. ☒☐ ARM Ltd.	
Peer review approval :	☒ Approved ☐ Rejected (improve as specified hereunder)	Date:	[26.09.2007]
Suggested improvements:			

Version History:

Version:	Comments, Changes, Status:	Person(s):
0.1	Introduction, tasks 2.3, 2.5, 2.6, results, conclusions	Aleksandra Dvoryanchikova
0.2	Task 2.2, results, conclusions, references	Rajasekaran Andiappan
0.3	Task 2.4, results, conclusions	Oscar Perez
0.4	Integration of different versions	Aleksandra Dvoryanchikova
0.5	Task 2.1	Bernard Bony
0.6 final	Minor changes over the whole text	Aleksandra Dvoryanchikova
Final	Assessment of the document	PCC members

[1] As defined in the DoW

[2] Scheduled date for approval

Table of Contents:

List of Figures:

List of Tables:

1. Introduction

The main objective of WP02 was to define an infrastructure framework for service-based ad hoc networking to enable communications between embedded devices at the application level. In order to provide semantic interoperability, the service descriptions were enriched with semantic content, enabling smart embedded devices to discover and invoke other devices that implement different taxonomies, syntax and/or morphology, but equivalent semantics. The framework also provides facilities for the management, administration, monitoring, and reconfiguring of the system.

An important aspect of this framework has been a facilitation of the integration of non-service-enabled devices by providing bridging or gateway mechanisms for embedded devices such as wireless sensors. In this way the device-centric infrastructure and the service-centric infrastructure of the SOCRADES project were combined.

The boundary between where devices can be fully service-enabled and where they cannot because of cost effectiveness, legacy preservation or other considerations and constraints must be kept variable, so that it can be shifted as technology makes further strides and device capabilities evolve accordingly.

The use of the Service-Oriented Architecture paradigm implemented through Web Services technologies, at the ad hoc device network level enabled the adoption of a unifying technology for all levels of the enterprise, from sensors and actuators to enterprise business processes. The benefits of service-orientation facilitate the discovery and composition of applications by re-configuration rather than re-programming. Dynamic self-configuration of smart embedded devices using loosely-coupled services has provided advantages for highly dynamic and ad hoc distributed applications, as opposed to the use of more rigid technologies such as those based on distributed objects.

In accordance to the chosen approach, the SOCRADES project was build atop the technology developed by the SIRENA project and extended it in several directions, including:

- Service composition, orchestration and co-ordination;
- Service management;
- Agent communication;
- Semantically enriched service description and discovery.

DPWS was employed as the technological foundation for implementing Web Services at the device level. A comprehensive framework for ad hoc device networking was specified on top of this foundation.

The implementation of the various framework elements will be carried out in WP5.

This milestone describes the key points of the applied approaches and technologies, provides with main results and potential problems and concludes the contributions.

2. Roadmap for reaching milestone 2.1

2.1. Task 2.1

Task 2.1 was started in M1 and completed M12 in accordance with the planned schedule.

The main objective of the task was to specify a device level service framework bringing the foundation for implementing Web Services at the device level. This framework includes the elements providing the basic DPWS functionalities, such as addressing, discovery, metadata exchange and eventing, which are already supported by the DPWS stack developed by the SIRENA project, but it must be enhanced and extended in order to support the specific requirements of the SOCRADES project and in particular the requirements coming from the other SOCRADES frameworks (service orchestration, service management…).

The main expected additions of the SOCRADES project to the device level service framework include:

- Adding an extensibility mechanism to allow new functionalities to be seamlessly integrated
- Adaptative Quality of Service support
- Dynamic service deployment
- Dynamic invocation of services
- Improving the discovery capabilities
- Improving performances and embeddability

The foundation of the proposed framework is to provide a modular and extensible architecture in which the main expected additions for the SOCRADES project can be either included as built-in functionalities or as additional and optional plug-ins. With this approach, the device level service framework can be scalable in order to fit in small devices with very limited resources. The extensibility mechanism is based on the identification of a set of extension points where the functions which extend the current DPWS specification can be easily plugged into the standard DPWS stack (see Figure 1).

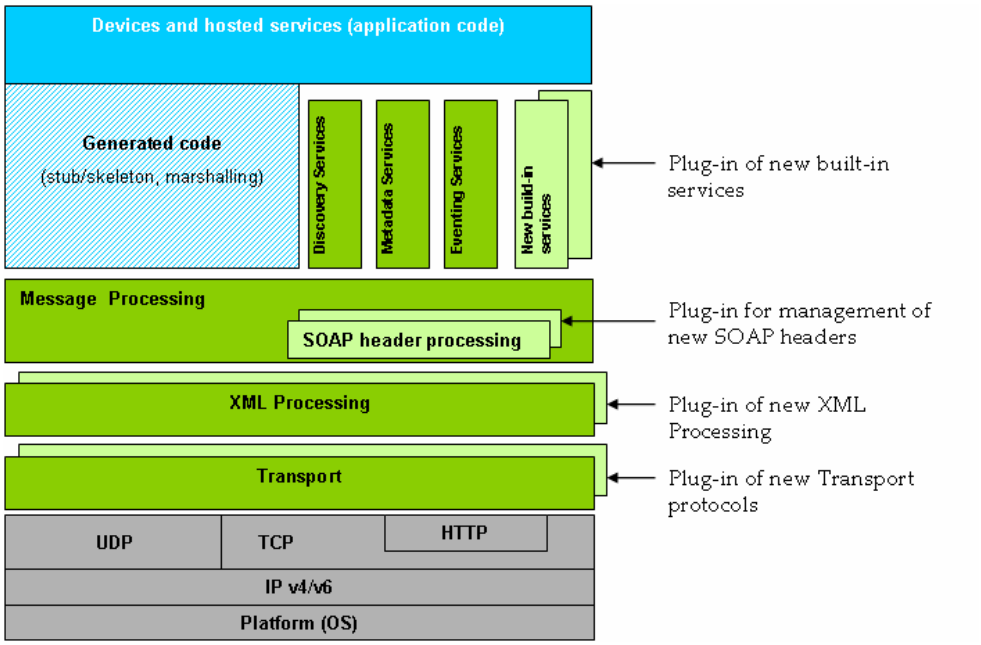

Figure 1: DPWS stack architecture with plug-in mechanism

With this new architecture, the same service implemented in a device can propose several Qualities of Service that a service consumer can discover and choose for invoking the service.

The standard behaviour of the current DPWS stack is that the services developed for a device are deployed statically at build-time. The SOCRADES device level service framework must implement a generic architecture that will support a dynamic deployment of the services. The purpose is to allow services, implemented either as native code or as specific interpreted code, to be deployed at runtime without stopping the already deployed services.

In very dynamic scenarios, client applications must be able to discover and use new and unknown services at runtime. The device level service framework will offer dynamic invocation capabilities, allowing generic client applications to call previously unknown services. In that case, the service invocation is not predefined at build-time with the help of code generation tools, but at runtime, based on the description of the discovered services.

The usual discovery mechanism proposed by the DPWS specification is limited inside one sub-network. In order to cover more complex applications, the device level service framework will offer advanced discovery capabilities for scenarios involving more complex network infrastructures, through the support and the extension of the so-called concept of “discovery proxy” mentioned in the DPWS specification.

The device level service framework must support to be embedded in small devices very limited in hardware resources. Moreover real-time performances must be achieved on these targets. Optimizations in size and performances will be provided in particular by using a scalable approach with for example a minimal version supporting the just enough built-in capabilities, and by supporting new XML processing technologies with high performances.

The results of the task are described in the internal task report and summarized in D2.1. The task was completed on schedule.

2.2. Task 2.2

Task 2.2 was started in M10 and in accordance with the planned schedule to be completed M18.

The service orchestration framework is responsible for the control and coordination of distributed services. It is not designed as a standalone framework, but a framework that is supported when conflict situations occur due to the high degree of flexibility.

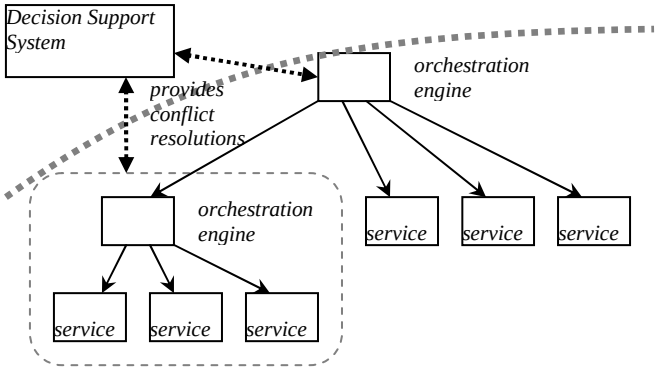

Figure 2: Control architecture of service-oriented systems coordinated by orchestration and supported by decision-making systems

The foundation of the proposed orchestration approach is to build reconfigurable automation systems of modular and simple automation components, each of which providing a set of services that represent their production capabilities (Figure 2). These automation components are mechanical devices and self-contained unit that are equipped with computing capabilities and control software for the logical control, with electrical interfaces to I/O and with a service interface exposing the internal functions as service.

An approach is described that uses Petri Net (PN) models

- to model, simulate, validate and analyze the behavior of components and the whole system virtually,
- to support the composition of the system from sub-models by utilization of appropriate methodologies in PN sub-models,
- potentially, to drive and orchestrate the run-time behavior of components.

It is assumed that the designing of the orchestration can take place in two different abstraction levels. A first level describes the behavioral part of the system that provides enough information to allow the application of analysis, simulation and validation methods. If a model representation of a system been found that is valid, analyzed, it can be the basis for further steps that refine the behavioral model to a more workflow-oriented model.

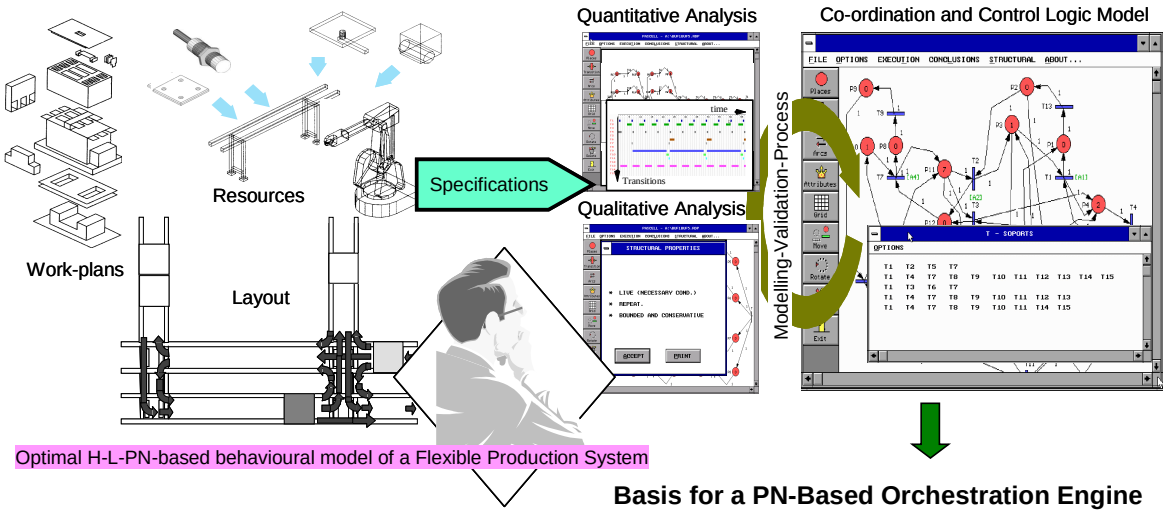

Figure 3: Approach modelling, validation and orchestration of FPS

The composition of the system from reusable components is supported by a 2D/3D engineering system. This way, the engineering tool facilitates the composition of PN sub-models into a complete system model, but it could also give input for other approaches. The validated synthesized model of the system will produce a so called orchestration engine which will be used to control the production system at runtime. This orchestration engine interprets an appropriate representation of the PN system model (Figure 3).

The results of the task are described in the draft of the internal task report and summarized in D2.1. Task implementation is on schedule.

2.3. Task 2.3

The task has started M12 and is to be completed M18.

The present task defines a framework for implementing management functionality usable across many application domains. A generic manageability framework will support such manageability functionality, which should be highly extensible so as to cover the broad spectrum of device complexity to be coped with. For higher-level devices, management functionality may go as far as to include provisions for reliability engineering, self-reconfiguring or self-healing. Over time, the sophistication barrier may gradually shift as embedded processing capabilities evolve.

At the time of submission of the SOCRADES proposal, the WS-Management specification was deemed to provide the most appropriate foundation for such generic management functionality, in particular because it is lightweight and therefore usable in resource-constrained devices. However, in the meantime, the OPC UA (Unified Architecture) specification was brought about by the OPC Foundation; OPC UA also defines a general-purpose management platform based on XML and Web Service Technology and offers functionality of a similar nature as that provided by WS-Management. Furthermore, OPC UA may be expected to garner significant attention in the automation field due to the established reputation and widespread use of previous OPC technology. Therefore, despite the fact that OPC UA was not part of the landscape initially addressed by SOCRADES, it was decided to start the activities of the present task by an in-depth comparison of DPS and WS-Management, on the one hand, and OPC UA on the other. This resulted in the very recent publication by Schneider Electric of a first version of a white paper entitled “DPWS vs. OPC UA”. The initial conclusions of this study are that:

- There is a definite advantage in attempting to integrate DPWS and OPC UA, given the large degree of commonality at the level of the Web Services protocols underlying both technologies.
- There is also an opportunity to make the resource model that is an integral part of OPC UA usable in the context of WS-Management and/or in the context of the future “converged” management standard,

tentatively referred to as WS-UnifiedManagement, destined to replace both WS-Management and WSDM (Web Services Distributed Management).

Further discussion will determine the path to be followed. This work is also highly relevant to WP6.

2.4. Task 2.4

Task 2.4 was started M9 and to be completed M18 in accordance with the planned schedule.

The need for a service-enabled agent framework is introduced in task 2.4. Two scenarios are envisioned, one where there is no service redundancy, i.e. a service is always provided by at most one device, and a more complex scenario where the same service may be provided for more than one device or by composition of devices.

The first scenario is the most basic scenario and thus completely addressable by merely a Service Oriented Architecture (SOA). However, in the second scenario there is need for a certain decision-making entity which is dependent on application-specific parameters. In this case, a facilitator technology must be introduced in order to address this scenario. The mapping service-device must be achieved in runtime since the decision is based on parameters that maximize a certain application specific need, i.e. shortest distance, maximum throughput, etc. Cases such as changes in the parameters during runtime or machine failure should be also captured while implementing the framework.

This decision-making entity only addresses complex scenarios where a fully SOA architecture is not sufficient and therefore should be seen as a facilitator for the SOA architecture. The most basic scenario, i.e. SOA architecture should be taken as a basis and most complex scenarios, i.e. where decision-making technology is needed, seen as independent extensions of the basic scenario.

Figure 4 introduces a system composed of 3 services, namely service A, B and C and 2 devices, X and Y. Device X provides service C, A and B whereas device Y provides service A. Service C is described as the sequence of first service A and then service B. A certain application A needs the service C which is composed during runtime with service A from device Y and service Y from device X. Subsequent calls may differ in the mapping service-device since the decision on how to make this mapping is based on parameters outside of the scope of the SOA which only should deal with services and the details on how to make the mapping hidden to the SOA architecture.

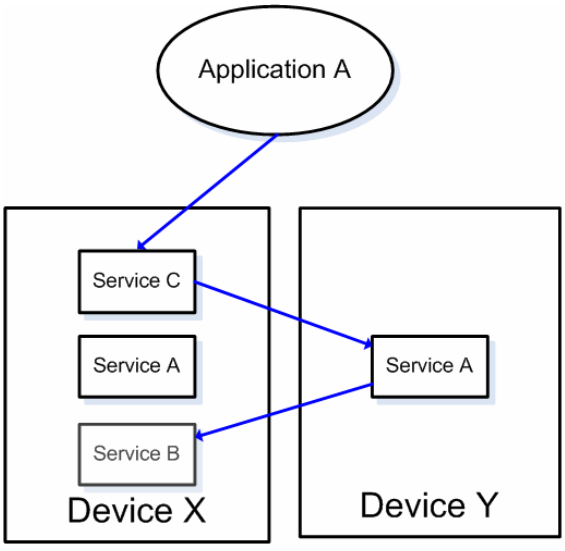

Figure 4: Composite service invocation with decision-making

The technology chosen for facilitating the decision-making task is Multiagent systems (MAS) due to their distributed nature, which allows easy reconfiguration. Furthermore, the messaging infrastructure provided by MAS, designated as agent communication language (ACL), bears a high degree of resemblance with the Web Service messaging infrastructure and thus the messaging channel can be made common to both technologies. Table 1 presents the message parameters specified in FIPA ACL and the corresponding message addressing parameters (MAPs) in WS-Addressing.

ACL parameter	Parameter category	WS-Addressing MAP
performative	Type of communicative act	Action
sender	Participant in communication	Source endpoint
receiver	Participant in communication	Destination endpoint
reply-to	Participant in communication	Reply endpoint, Fault endpoint
protocol	Control of conversation	
conversation-id	Control of conversation	Message id
reply-with	Control of conversation	Message id
in-reply-to	Control of conversation	Relationship
reply-by	Control of conversation	Message id
language	Description of Content	
encoding	Description of Content	
ontology	Description of Content	
content	Content of message	

Table 1: FIPA ACL message parameters

Regarding the integration of agent technology in the devices, there are many possibilities on how this can be achieved. The most neutral solution would be to introduce an agent per each device. A web service can be used in the cases where more information is needed from the agent in order to make an optimal solution. The agent would query the web service asking from more information from a certain device and hence to be able to reach a decision. This could also be implemented using the mobility feature of the agents. Agents can move within devices as long as the agent container is common in all the devices. However, this is a risk since there would be a high dependence of the chosen agent platform and it is preferable to deploy a neutral framework where each vendor could use its own implementation while assuring seamless integration among them.

The results of the task are described in the draft of the internal task report and summarized in D2.1. Task implementation is on schedule.

2.5. Task 2.5

Task 2.5 was started M4, and completed and reported M12 in accordance to the planned schedule.

In order to develop the Semantic Web Services framework an interdisciplinary approach was introduced, to be numbered the involved area of knowledge but a few: ontology and cognitive science, software architecture, service-oriented architecture and Web Services, distributed event-based systems, multi-agent systems and holonic manufacturing systems and other. Knowledge-based approach was introduced to system configuration and event-based approach was proposed for cognition. A set of middleware tools and methods was applied in order to provide an infrastructure for developing manufacturing systems and incorporate the qualitative attributes like integrability, convertability, adaptability, reusability, interoperability, scalability, and evolvability.

The ontology components were introduced to provide semantic descriptions. The world model consists of four parts: process, product, equipment and service upper ontologies. The ontologies for processes, manufacturing equipment and products are defined using the OWL-DL ontology language (Figure 5), which provides SHOIN(D) Description Logics semantics. The OWL-S upper ontology is used for describing services and their underlying processes are presented. For defining conditional and effect axioms that cannot be expressed using DL, the Semantic Web Rule Language (SWRL) is used.

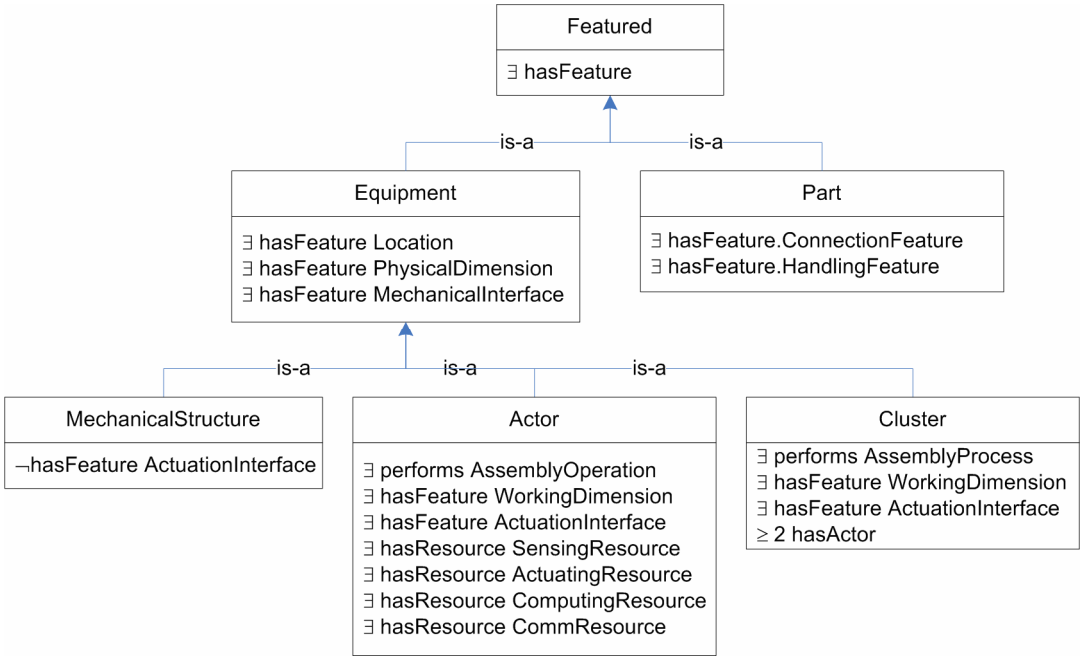

Figure 5: Summary of Equipment concepts and properties

At the proposed information infrastructure, services are associated to semantic descriptions in order to facilitate their selection and invocation after dynamic discovery (Figure 6). The information level of abstraction provides the number of functions: service deployment, service discovery, service engagement, service invocation, world model acquisition, condition notification, and precondition and effect enrichment. The functions provided at this higher level of abstraction, which build on the primitive interaction types provided by DPWS.

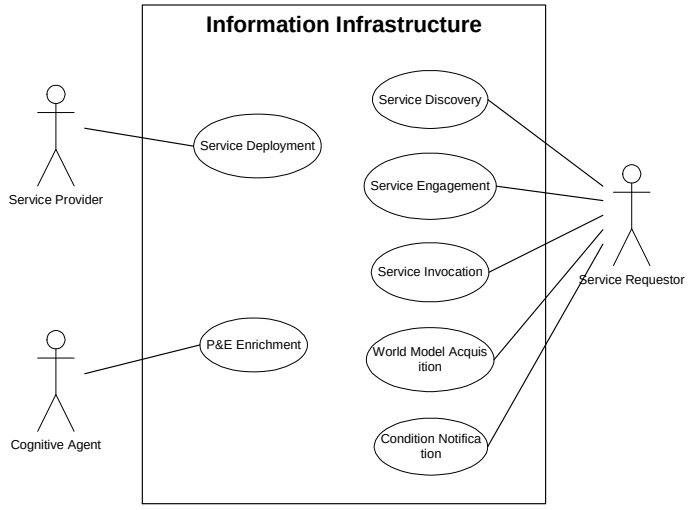

Figure 6: Use case diagram for the Information Infrastructure middleware functions

The proposed middleware functions and abstractions were tested in a testbed.

The task results were described in details in the internal task report and summarized for the deliverable D2.1. The task was completed and reported at time.

2.6. Task 2.6

Task 2.6 was started M1, and completed and reported M8 in accordance to the planned schedule.

In order to raise the applicability of service-enabled devices there was proposed a service-oriented gateway approach. A gateway being an embedded device that is readily integrated into the SOCRADES infrastructure and that is also able to interact with non service-enabled devices. Therefore, gateways provide two interfaces: a Web Service interface to the SOCRADES infrastructure, and a *back-end* interface using other protocols to interface to other devices/protocols/infrastructures.

Figure 7 gives a schematic representation of gateway based communication between two networks. It shows a gateway between a SOCRADES network and a non-service enabled device network.

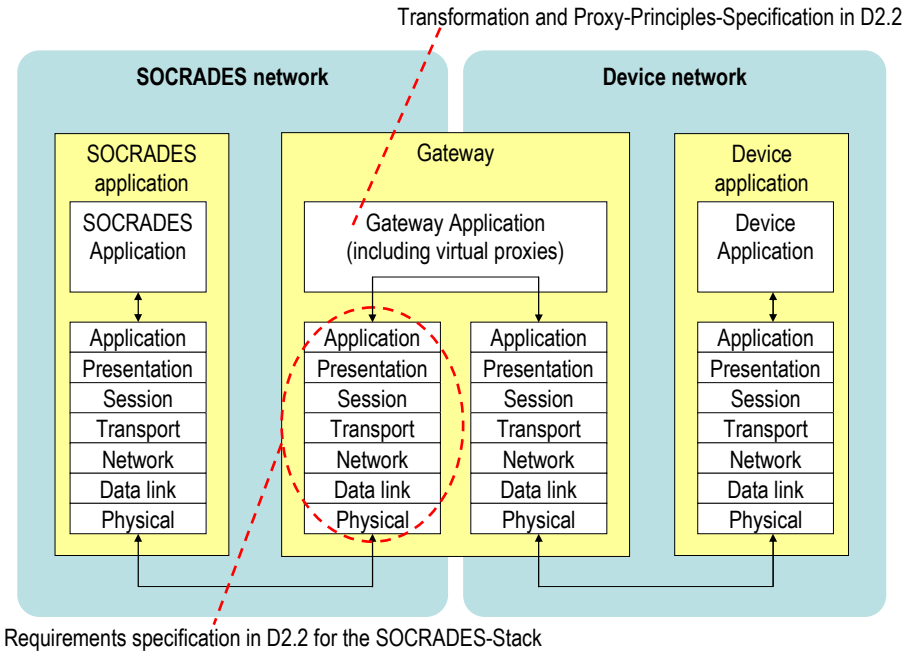

Figure 7: SOCRADES gateway principle

In general, abstraction from device specific representations of properties and data is recommended. This allows usage of generic mechanisms and services in the SOCRADES network. If required, additional device specific services may be offered by the gateway. For instance, a SOCRADES application may only be interested in a certain representation of data delivered by a wireless sensor network regardless of the identity and attributes of the specific sensor nodes that delivered those data. This may be achieved by providing data centric abstractions of the wireless sensor network that hide unnecessary details. For administration and maintenance purposes, on the other hand, those details may be provided via a specific sensor node management service.

The SOCRADES infrastructure proposes two different types of gateway devices: *gateways*, which are intended to provide transport of a legacy protocol using Web Services; and *mediators*, which are intended to provide a service-oriented interface to processes implemented using legacy protocols.

Table 2 summarizes the characteristics and differences between gateways and mediators. Use cases that follow in the next subsections will further illustrate scenarios in which either type of device is used.

Device type	Front end protocol	Back end protocol	Application model
Gateway	Web Services	Legacy protocol	Legacy
Mediator	Web Services	Legacy protocol	Web Services

Table 2: Summary of gateway and mediator characteristics

The approach was applied for a set of cases.

The task's results were described in detail in the deliverable D2.2. The task was completed and reported on schedule.

3. Problems and common results

The necessity of having decision-making and conflict resolution at runtime is addressed, because a system model does not describe a fixed sequence of actions, but rather all possible combinations thereof. For instance, if a system is flexible through redundancy of service providers, the concrete mapping of services to devices has to be decided at runtime. Two different scenarios have been identified, i.e. those needing decision-making and those where a fully SOA architecture is sufficient. The most basic scenario should be seen as the basis of the system whereas the scenario needing decision-making an extension of the former. Therefore, it is proposed to have the layer of system orchestration separated from the decision-making layer. The orchestration engine could be either distributed or centralized. The centralized approach seems more feasible since having a distributed orchestration engine in every device may not be a feasible solution in a resource constrained device.

The proposed infrastructure for Semantic Web Services provides the fundamental groundwork for achieving the set of objectives. However, it is envisioned that there is a room for additional functionality to be introduced to the infrastructure. For instance, currently it is possible for two concurrent software elements to simultaneously invoke services that affect an overlapping domain; the synchronization of the invocations is currently responsibility of the application level.

4. Conclusion

The service orchestration framework is responsible for the control and coordination of distributed services. The approach dealt with in this task applies to systems with a high necessity of dynamicity and flexibility, hence production cell or line level. A service orchestration framework that is highly modular and covers all potential scenarios has been introduced. The feasibility of this framework is yet to be tested but the risks have been thoroughly analyzed and different possibilities have been proposed in order to overcome the technical problems that the actual implementation of these frameworks may pose.

For the definition of the service manageability framework, the newly emerging OPC UA standard is being taken into account.

An approach to semantically describe processes was introduced, together with their requirements for invocation and their effects in the world, as well as a mechanism to facilitate cognition in the absence of sensorial information by propagating events that are associated to explicit description of their effects on the world.

A gateway architecture was specified, using the approach of device and virtual machine proxies; the description of the use case of a PC-based SOCRADES gateway including device proxies and virtual machine proxies; and the state machines for the PC-based SOCRADES gateway. An FPGA platform was introduced as the most cost effective method to build a flexible design platform. The preliminary specification for the platform along with the associated resource plan has been prepared.

Concluding, up to the present milestone, three of six tasks were completed and reported on time and three others have been started and are progressing in accordance with the planned SOCRADES schedule.