Towards A Generic, Service-Oriented Framework for Distributed Real-Time Systems

Muhammad Umer Tariq[1], Mohammad Abdullah Al Faruque[2], Santiago Grijalva[3], Marilyn Wolf[4]

[1,3,4]*Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA.*
[1]m.umer.tariq@gatech.edu
[3,4]{sgrijalva, wolf}@ece.gatech.edu

[2]*Department of Electrical Engineering and Computer Science, University of California, Irvine, USA.*
[2]mohammad.alfaruque@uci.edu

***Abstract*—Continuously increasing complexity and scale of distributed real-time systems have exposed the limitations of their existing development methodologies. This fact is evident by the unsustainable rate of increase in the development and maintenance costs of such systems. In this paper, we present a generic, service-oriented framework for distributed real-time systems. The proposed framework can potentially serve as the basis for a widely applicable, cross-domain toolset, thus, decreasing the development and maintenance costs for distributed real-time systems. The proposed framework consists of a generic, service-oriented deployment platform that abstracts away the details of implementation platform and an associated development methodology. The proposed framework makes extensive use of the existing service-oriented technologies such as Web Services. However, it also extends these technologies for application to distributed real-time systems by introducing QoS-aware service deployment and service monitoring phases. This paper presents the details of the proposed framework as well as a case-study of the application of the proposed framework to the domain of smart grid.**

Keywords-real-time systems; service-oriented computing; smart grid; cyber-physical systems; web services

I. INTRODUCTION

Continuously increasing complexity of distributed real-time systems has exposed the limitations of existing development techniques in tackling this complexity. Because of these limitations, development and maintenance costs for these systems are increasing at an unsustainable rate. Moreover, widespread availability of economical communication and computation infrastructure has enabled the emergence of a new category of wide-area real-time systems such as smart power grid [1] and vehicular networks [2]. The scale of these systems cannot be easily managed by existing set of development techniques for distributed real-time systems. In order to successfully handle the increased complexity and scale of distributed real-time systems, a new set of design techniques and associated tools are required. However, there is a wide variety of distributed real-time systems with varying geographic scale and varying strictness of timing constraints. Designing a domain-specific toolset to handle each of these examples of distributed real-time systems will be highly inefficient [3] [4]. This has created interest in a unifying, generic framework for distributed real-time systems that can form the basis of highly effective, cross-domain design methodologies and toolsets [5]. For instance, such a framework could serve as the foundation of a widely applicable model-driven toolset for distributed real-time systems.

Over the last decade, service-oriented computing (SOC) paradigm [6] has seen considerable success in the field of enterprise integration and electronic commerce. SOC paradigm divides application developers into three distinct roles: service providers, service consumers, and service brokers. Service providers implement, describe, and publish their services in the directories maintained by service brokers. Service consumers discover the required services by contacting the service broker. Once service consumers have discovered a service, they can interact directly with the service providers according to the service description. SOC enables loose coupling between the components of the systems. This loose coupling of components results in creating systems that are more flexible and manageable.

Traditional distributed, real-time systems such as avionics and automotive systems did not follow a service-oriented approach. However, the growing complexity of distributed real-time systems and the emergence of wide-area real-time systems such as smart power grid are rendering traditional real-time system design methodologies ineffective. Service-oriented computing presents a promising approach to handle the growing complexity and scale of real-time systems. However, the service-oriented computing techniques standardized in the field of enterprise integration applications cannot be directly applied to real-time applications. In the recent past, efforts have been made to apply service-oriented computing to the domain of industrial automation. These efforts have resulted in a standard known as Device Profile for Web Services (DPWS), which brings the service-oriented computing concepts from enterprise computing to the world of resource-constrained embedded devices [7]. However, the focus of these efforts has been the interoperability of networked embedded devices rather than provision of mechanisms that support the use of service-oriented computing in real-time systems with hard timing constraints. Moreover, these efforts do not address the need for a generic framework that can capture the application of service-oriented computing to the whole spectrum of real-time systems with varying scales and varying strictness of timing constraints.

In this paper, we present a service-oriented framework for distributed real-time system that is generic enough to be applied to real-time systems of varying scales and multiple domains. Due to its universal and generic nature, it can serve as the basis for widely applicable design techniques and tools. Moreover, it can serve as the common foundation of model-driven development and operation toolsets for various diverse application domains such as smart power grid, smart irrigation networks, and robotics. We present the key entities and XML documents involved in our framework. We explain the proposed framework through a case study of its application to a canonical smart grid scenario.

The proposed framework does not just represent a direct translation of service-oriented architecture concepts (service description, service publication, service discovery, service binding and service interaction) from enterprise computing domain to real-time systems domain. Actually, the framework is much more fundamental and represents a unified approach to tackle various types of distributed real-time systems. The framework considers service as the basic modeling entity. The framework proposes a resource-aware service deployment step and supports QoS-aware interaction of various services.

The rest of the paper is organized as follows. Section II outlines some related work. Section III presents an overview of the traditional service-oriented computing that utilizes a set of standards known as Web Services. Section IV describes the set of unique challenges faced in the application of service-oriented computing to distributed real-time systems. Section V presents the details of the proposed framework. Section VI presents a case-study for the application of the proposed framework to distributed real-time systems. Section VII outlines some of the advantages of the proposed framework. Section VIII presents the conclusion and future work.

II. Related Work

A cross-domain framework to model the behavior of real-time computer systems has been the focus of much attention recently because of its potential for providing a set of widely applicable tools that can help tackle the problem of continuously rising development and maintenance costs of embedded real-time systems. Reference [5] presents a model of the behavior of real-time system based on three basic concepts of computational component, state and message. Our proposed framework is not in conflict with these concepts. Actually, it builds on these concepts and introduces the concept of service as the central entity. This allows our proposed framework to incorporate the rich set of concepts available in the domain of service-oriented computing into the development of real-time systems.

A number of recent research projects have explored the application of SOC paradigm to distributed real-time systems [8][9][10]. These efforts have focused on porting Web Service technologies to resource-constraint devices, resulting in Device Profile for Web Services (DPWS) [7]. These efforts do not concentrate on enhancing the traditional SOC paradigm with mechanisms that will allow its application to real-time systems with timing constraints. Moreover, for the most part, these efforts have been focused on the domain of industrial automation. Our proposed service-oriented framework tries to incorporate the contribution made by these projects but it also aims to be more generic and widely applicable in nature. Our proposed service-oriented framework for distributed real-time systems is based on certain enhancements to SOC paradigm (such as resource-aware service deployment and QoS-aware service monitoring) that will allow it to handle the complete lifecycle of a wide-range of distributed real-time systems with varying level of strictness in their timing constraints.

QoS-aware service-oriented computing has been a topic of interest in the traditional service-oriented computing research community. For instance, [11] presents WS-QoS framework that enables QoS-aware service discovery and service monitoring. WS-QoS framework is not directly applicable to the development of embedded real-time systems as it does not tackle the service deployment on a platform with limited resources. Our proposed framework overcomes this deficiency of WS-QoS framework by providing a QoS-aware service deployment step.

III. Traditional Service-Oriented Computing

Traditionally, service-oriented computing has been applied for enterprise integration applications. Various standardization efforts have played a significant role in the successful implementation of service-oriented computing paradigm. These efforts have resulted in a set of standards known as Web Services that deal with the three major aspects of service-oriented computing [6].

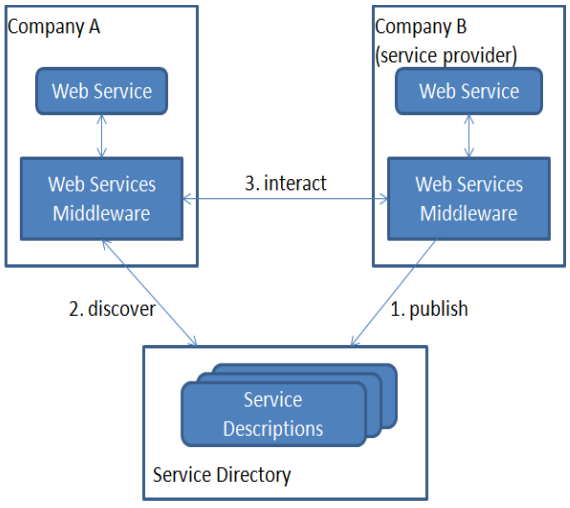

Figure 1. Service-oriented computing through Web Services.

A. *Service Description*

Web Services Description Language (WSDL) is the standard that deals with service description. WSDL defines the syntax of XML documents that describe Web Services. WSDL document defines the interface of a service as a set of input and output messages.

B. *Service Discovery*

Universal Description Discovery and Integration (UDDI) is the standard that deals with the service discovery process. UDDI is based on a *business registry*, which is essentially a directory of available services. UDDI provides an API for publishing service descriptions in the registry and querying the registry for required services.

C. *Service Interaction*

Simple Object Access Protocol (SOAP) provides a message format that can be used by a service to exchange messages with another service. SOAP message consists of an envelope, where each envelope contains two parts: a header and a body. SOAP message acts as a generic conveyer of information between services. In the case of Web Services, SOAP messages are generally transported using HTTP. However, a binding could be defined to other transport protocols as well. A binding defines how a SOAP message is transported from one service to another using the transport protocol primitives.

IV. Service-Oriented Computing for Real-Time Systems

Service-oriented computing has been successfully applied in business enterprise integration applications using the Web Services standard. However, real-time systems have certain distinct characteristics that are not handled by the existing standards and technologies available for service-oriented computing paradigm. In this section, we present a set of additional capabilities that are needed to successfully apply SOC paradigm to distributed real-time applications.

A. *Service Description*

For the application of service-oriented computing to real-time systems, service must be treated as a modeling entity. Service description should not only include its interface in terms of messages that it can receive and send, but it should also include QoS properties of these messages as well as the platform resources required to deploy a service on a certain platform.

B. *Service Deployment*

Service deployment capability should be able to read in QoS- and resource-aware service description and decide whether it can be deployed on a certain computing platform. Once the deployment decision has been made based on the service description and resources available on a platform, there should be a mechanism to actually deploy a service on the platform. This mechanism could be off-line or on-line depending on the type of the real-time system. This mechanism will involve transferring the service logic or service code on to the platform and provisioning the required communication and computation resources of the platform to the service.

C. *Service Discovery*

Real-time systems require QoS-aware service discovery. Service providers and service consumers should not only agree on the messages forming the interface between them but also the timing properties associated with these messages.

D. *Service Monitoring*

When the services have been deployed and they start exchanging messages with each other through a communication medium, a run-time facility must continuously monitor these message exchanges. This monitoring capability can result in the timely detection of violations of QoS constraints for different services as a result of some failure or excessive delays in the communication network. This detection capability can be used to notify the application software of a QoS failure, allowing the application software to take a mitigating action based on the nature of real-time application.

V. Proposed Service-Oriented Framework

In this section, we present the details of our proposed framework that can provide the capabilities discussed in the last section and thus, enable the application of SOC paradigm to real-time systems. The proposed framework consists of a generic service-oriented platform and an associated development methodology. Our proposed framework makes the maximum use of the existing standards and technologies available as a part of the Web Services. However, it extends some of these technologies to provide the capabilities discussed in the last section. For instance, this framework extends traditional Web Services framework by adding a deployment proxy and a QoS proxy to ensure that platform resources are not overloaded and QoS failures are detected in a timely manner.

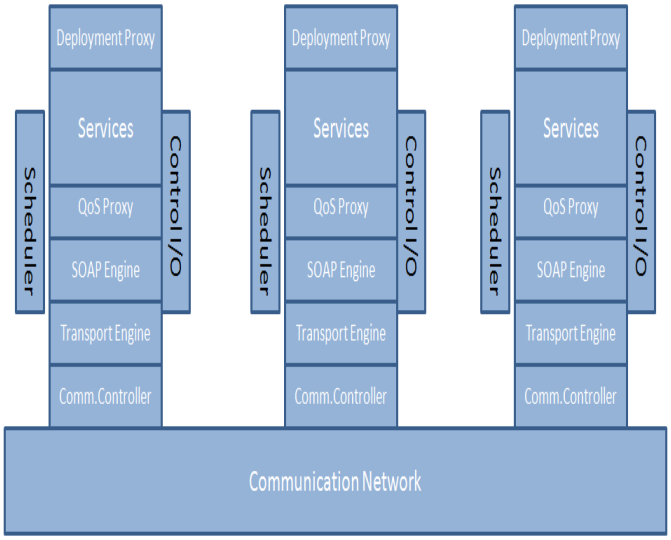

Figure 2. A generic, service-oriented platform for distributed real-time systems.

A. *Generic Service-Oriented Platform*

The proposed generic service-oriented platform abstracts away the heterogeneous implementation platforms used for distributed real-time systems and presents a generic

interface for development of service-oriented real-time systems. Fig. 2 presents the proposed generic service-oriented platform. The proposed platform defines two types of entities: platform components and XML documents. Following are the major platform components: *Deployment Proxy, QoS Proxy, SOAP Engine, Transport Engine, Communication Controller, Scheduler, Control I/O*. Following are some of the XML documents required: *PlatformRequirementDocument, PlatformOfferDocument, QoSOfferDocument, QoSRequirementDocument.*

1) XML-based Documents

PlatformRequirementDocument defines the platform resources that a service needs in order to be successfully deployed on a platform. *PlatformOfferDocument* defines the currently free resources that are available on a certain platform. *QoSOfferDocument* defines various QoS aspects that a service offers to provide to its consumers. *QoSRequiremetDocument* defines the QoS requirements that a service needs from another service.

2) Platform Components

Deployment Proxy provides the interface that is used by the administrator to deploy a new service on the proposed service-oriented platform. Administrator contacts the *Deployment Proxy* with *PlatformRequirementDocument* of a service. *Deployment Proxy* consults the *PlatformOfferDocument* of the platform and decides whether the service can be deployed on the platform. Once it has been decided that there are enough resources to deploy a service, *Development Proxy* follows a protocol to transfer the code for the service on to the platform.

QoS Proxy monitors the messages exchanged by the service and informs the service in a well-defined and timely manner of any violations of constraints identified in *QoSRequirementDocument* and *QoSOfferDocument*. Our proposed platform allows services to exchange SOAP messages. *SOAP Engine* takes care of binding SOAP messages to lower-level transport protocol. *Transport Engine* handles the processing related to a certain networking technology such as TCP, UDP or some other similar transport protocol.

The proposed platform has to meet various real-time constraints on its processes. Therefore, the platform must be built over a real-time computing base. *Scheduler* represents the real-time operating system facilities that enable the timely processing of various tasks in the proposed service-oriented platform. *Control I/O* block allows the platform to do the sensing and control actions in order to control a physical process. This capability allows the proposed service-oriented platform to participate in a control system.

B. Service-Oriented Development Methodology

The proposed service-oriented framework for distributed real-time systems identifies a development methodology that makes use of the service-oriented platform described in the last section. Fig. 3 shows the main steps involved in the proposed service-oriented development methodology.

The platform porting step involves abstracting away an implementation platform by porting the generic service-oriented platform to that particular implementation platform. In the service modelling step, the distributed real-time system is modelled as a set of interacting services. The requirements of the real-time system drive this modelling process. In the service implementation phase, the code for the services is developed. In the service description step, concrete XML-based documents are defined. In the service deployment phase, an appropriate platform is identified that can provide sufficient resources to the service and the service is deployed on it through its *Deployment Proxy*. In the service discovery phase, each service publishes itself to a directory and these directories are used by services to discover their partner services. Both the service deployment and service discovery phases could occur off-line or online depending on the nature of distributed real-time application.

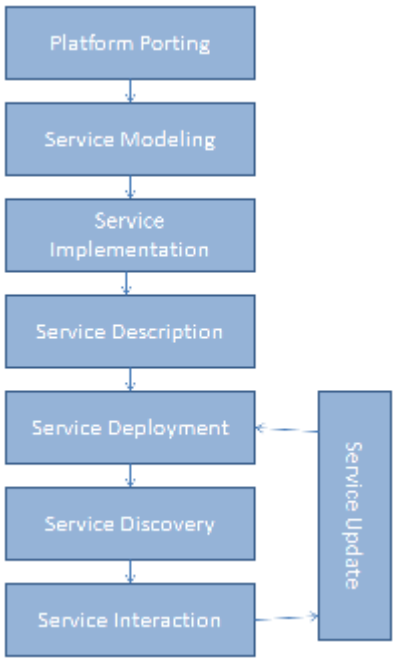

Figure 3. A generic service-oriented methodology for real-time systems.

In the service interaction phase, each service sends a SOAP message. SOAP message passes through the *QoS Proxy* to check for QoS violations. *SOAP Engine* deals with transforming the SOAP messages to the underlying transport protocols available on the implementation platform. On the receiving end, the message is again checked by the *QoS Proxy* for any QoS violations. In case a message is not received in time, *QoS Proxy* informs the service about this delay through an exception or a similar mechanism. This allows the service to take appropriate action if needed. This capability allows the proposed service-oriented framework to be used in a closed-loop control system with unpredictable network delays. The service update phase exists to help in upgrading and maintaining an existing distributed real-time system. In the service update phase, modifications are made to the service

descriptions and service logic and the service deployment phase is repeated to re-deploy such services on the implementation platform.

VI. CASE STUDY

This section explains the ideas presented in this paper through a case study of applying the proposed service-oriented framework to a simplified but canonical scenario of smart grid. The scenario consists of a power system with a generator, a load and a controller. In the absence of storage capacity in a power system, generation must match the load at all times to keep the frequency stable and to ensure the safe operation of power system [12]. Usually, the load changes according to the demands put on the system by the user and the generation must keep changing to match the load. However, certain renewable energy sources such as wind and solar are intermittent in nature because of the natural variations in the wind and solar intensity. One approach for maintaining the generation-load balance in the presence of intermittent generation resources is to continuously update the controller about the generation level of an intermittent generation resource, which in turn sends a signal to a load asking it to change its consumption level.

We consider the development of such an application through our proposed service-oriented framework for distributed real-time systems. We assume that there is a computation node associated with each of the components (generator, load, and controller) and these computation nodes are connected to each other through a communication network. We also assume that through a process of platform porting, each computation node supports the proposed generic service-oriented platform. In the service modelling step, we identify the following services: *genUpdateService*, *loadControlService*, and *controlCenterService*. *genUpdateService* sends the generation level of the generator to the *controlCenterService* every T seconds. The *controlCenterService* receives the generator status from *genUpdateService* every T seconds and sends out a message to the *loadControlService* every T second. In case the *controlCenterService* does not receive a message in time from the *genUpdateService*, it prepares an appropriate message for the *laodControlService*. *LoadControlService* receives a message from the *controlCenterService* every T seconds and changes the power consumption level appropriately. In case the consumption level cannot be changed, the *loadControlService* sends a response back to the *controlCenterService*.

In the service implementation phase, code is developed for each of the three services in an appropriate development environment. In the service description phase, XML documents such as *ServiceDescriptionDocument*, *PlatformRequirementDocument*, *QoSOfferDocument* and *QoSRequirementDocument* are developed for each of the three services. In service deployment phase, each of the service is deployed on the appropriate computation node through the corresponding *Deployment Proxy*. Once the services have been deployed, they publish themselves to an appropriate directory. In this case study, control centre could be an appropriate location for the service directory. Services also try to discover their partner services using the appropriate directory. Once, the services have discovered each other, they start sending SOAP messages to one another. In this service interaction phase, *QoS Proxy*, at each computation node, informs the service in a well-defined timely manner if an expected message is not received according to the *QoSRequirementDocument*. This allows the service to take appropriate remedial measures. For instance, in the case of *controlCenterService*, when the message from *genUpdateService* does not arrive in time, *controlCentreService* uses other control centre capabilities (such as state estimation) to generate an appropriate message to be sent out to the *loadControlService*.

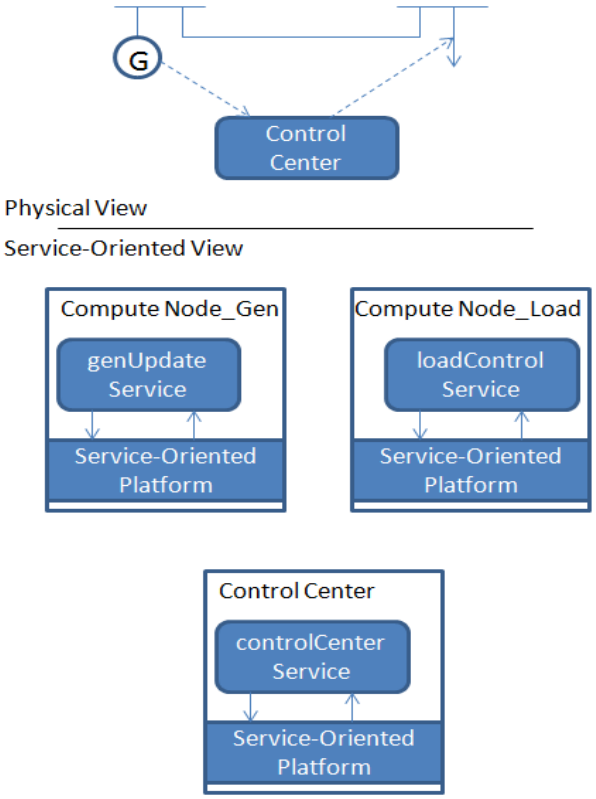

Figure 4. Case study.

Now, let us assume that the intermittent generation resource has some associated storage capability that allows extra energy generated to be stored and used later. In order to control this resource, we shall need to develop a new service named *storageControlService*. We go through the service modeling, service implementation, and service description phases for this new service. Let us assume that this service will be deployed on the same computation node, which contains the *genUpdateService*. Then, in the service deployment phase, *Deployment Proxy* of the corresponding computation node will make sure that *PlatformRequirementDocument* of *storageControlService* and *PlatformOfferDocument* of the corresponding computation node are compatible with each other. If enough resources are available, the new service will be deployed on

the same computation node that contains the *genUpdateService*.

VII. Advantages of the Proposed Framework

In this section, we present the various advantages that can be obtained by utilizing the proposed generic, service-oriented framework.

A. Cross-Domain Tools

The proposed generic, service-oriented framework can acts as the foundation for a set of tools (such as model-driven toolsets) that can be applied across multiple domains of distributed real-time systems. The wide applicability of such a toolset will ensure that sufficient investment is made in improving the quality of this toolset without increasing the cost of such a toolset.

B. Support for Closed-Loop Control Systems

The proposed framework helps to reduce the unpredictability in the delays faced by control messages and also supports the timely notification of the failure of the communication network in delivering an expected control message. This allows the use of this framework in development of wide-area, closed-loop control systems.

C. Better Lifecycle Management

The proposed framework supports a resource-aware deployment of a service on a certain implementation platform. This capability is really useful in upgrading the deployed systems as new services can be deployed with the confidence that they would not interfere with the correct operation of existing services.

VIII. Conclusion and Future Work

We have proposed a generic, service-oriented framework for distributed real-time systems. The proposed framework builds on the service-oriented technologies available in the enterprise computing domain and suggests certain extension to these technologies that can make them effective for the domain of real-time systems. The proposed framework can serve as the basis of a widely applicable, cross-domain toolset for design, development, and operation of distributed real-time systems. Such a toolset can be really useful for the cost-effective development of reliable and maintainable distributed real-time systems.

In the future, we plan to further explore the ideas presented in this paper through an ns-3 based simulation infrastructure [13]. This simulation infrastructure will extend the core ns-3 simulator with a middleware object that mimics the generic service-oriented platform presented in this paper. This will be followed by the RTOS-based implementation of the service-oriented platform proposed in this paper.

References

[1] K. Tomosovic, D.E. Bakken, V. Venkatasubramanian and A. Bose, "Designing the Next Generation of Real Time Control, Communication and Computation for Large Power Systems," *Proceeding of the IEEE*, vol. 93, no. 5, pp.965-979, May 2005.

[2] H. Moustafa and Y. Zhang, Vehicular Networks: Techniques, Standards, and Applications. Auerbach Publications, 2005.

[3] M.U. Tariq, S. Grijalva, and M. Wolf, "Towards a Distributed, Service-Oriented Control Infrastructure for Smart Grid," Proc. IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS 11), IEEE Press, 2011, pp. 35-44, doi:10.1109/ICCPS.2011.16.

[4] M.U. Tariq, H. A. Nasir, A. Muhammad, and M. Wolf, "Model-Driven Performance Analysis of Large Scale Irrigation Networks," Proc. IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS 12), IEEE Press, 2012, pp. 151-160, doi:10.1109/ICCPS.2012.23.

[5] H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, 2nd ed. New York: Springer, 2011.

[6] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design. New Jersey: Prentice Hall, 2005.

[7] (2009) OASIS website. [Online]. Available: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html

[8] M. Garcia-Valls, I. Rodriguez-Lopez, and L. Fernandez-Villar, "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Service-Oriented Distributed Real-Time Systems," IEEE Transactions on Industrial Informatics, vol. PP, May 2012, pp 1-1, doi:10.1109/TII.2012.2198662.

[9] (2009) SIRENA website. [Online]. Available: http://www.sirena-itea.org/

[10] (2010) SOCRADES website. [Online]. Available: http://www.socrades.eu

[11] M. Tian, A. Gramm, H. Ritter, and J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework," Proc. IEEE/WIC/ACM International Conference on Web Intelligence (WI 04), IEEE Press, 2004, pp. 152-158, doi:10.1109/WI.2004.10084.

[12] P. Kundur, Power System Stability and Control. New York: McGraW-Hill, 1994.

[13] (2012) ns-3 website. [Online]. Available: http://www.nsnam.org/