A SERVICE-OREINTED ARCHITECTURE FOR STRUCTURAL HEALTH MONITORING USING SMART SENSORS

J.A. Rice,[1] K.A. Mechitov,[2] B.F. Spencer, Jr.[3] and G.A. Agha[4]

[1] *Doctoral Candidate, Dept. of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, USA*
[2] *Doctoral Candidate, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA*
[3] *Professor, Dept. of Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, USA*
[4] *Professor, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA*
Email: jarice@uiuc.edu, mechitov@cs.uiuc.edu, bfs@uiuc.edu, agha@cs.uiuc.edu

ABSTRACT :

Structural Health Monitoring (SHM) can be an important tool for integrity assessment of structures after extreme loading conditions, as well for ongoing maintenance of aging infrastructure. SHM gives insight into a structures' response during an earthquake event which can in turn provide valuable information on the post-earthquake condition of the structure. The ultimate goals of implementing an SHM system are to improve infrastructure maintenance, increase public safety, and minimize the economic impact of an earthquake or other extreme loading event by streamlining repair and retrofit measures. Networks of wireless, smart sensors offer promise for accurate and continuous structural monitoring using a dense array of inexpensive sensors. Acquiring pertinent information from these networks is accomplished by leveraging the onboard processing capability of the sensor nodes to implement distributed damage detection algorithms. One of the major roadblocks to achieving such a system is the magnitude and complexity of the required software development. Service-oriented architecture (SOA) can significantly simplify this process by offering a modular, reusable and extensible approach to software development. Based on this approach, we have created an open-source toolkit for the development of SHM systems deployed on smart sensors that is available for broad use. This software facilitates the development of a variety of SHM applications using smart sensors and can be further expanded for other similar challenges.

KEYWORDS: Structural health monitoring, smart sensors, wireless sensors, service-oriented architecture.

1. INTRODUCTION

Structural health monitoring (SHM) provides the means for capturing structural response and assessing structural condition for a variety of purposes. For example, the information from an SHM system can be used to fine-tune idealized structural models, thereby allowing more accurate prediction of the response due to extreme loading conditions, such as an earthquake (Farrar and Doebling 1997). SHM also provides the characterization of loads in situ, which can allow the detection of unusual loading conditions as well as validate the structure's design. In addition, real-time monitoring systems can measure the response of a structure before, during and after a strong motion event, and used in damage detection algorithms assess the post-earthquake condition of a structure.

Given the size and complexity of many civil structures, a large network of sensors is required to adequately assess the structural condition. Traditional structural monitoring systems have been moving in the direction of dense deployment in recent years; however, the cost of installation can be thousands of dollars per sensor channel, and the amount of data generated by such a system can render the problem intractable (Nagayama et al 2007a). Networks of wireless smart sensors have the potential to improve SHM dramatically by allowing for dense networks of sensors to be installed on a structure (Spencer et al 2004, Gao and Spencer 2008).

Recognizing that damage in structures is a local phenomenon, SHM applications using smart sensors may be realized even for structures of substantial size. Responses from sensors close to the damaged site are

expected to be more heavily influenced than those far from the damage. The on-board computational ability of smart sensors allows data to be locally processed, thus maintaining reasonable communication requirements, preserving network resources and avoiding data inundation (Nagayama and Spencer 2007).

Though the application of smart sensors has been demonstrated (Kim et al 2007; Lynch and Loh 2006), there are several hurdles that have limited their widespread use for structural monitoring. In general, critical issues inherent in smart sensor networks (SSNs), such as synchronized sensing and data loss (Nagayama et al 2007b), still need to be addressed. In addition, the numerical algorithms required for system identification and damage detection must be implemented on sensor nodes which have limited resources. The result is that SHM applications require complex programming, ranging from network functionality to algorithm implementation. Applications software development is made even more difficult by the fact that many smart sensor platforms employ unique operating systems which are implemented in unfamiliar programming languages. Requiring that engineers interested in deploying wireless SHM systems have the expertise to develop the complex software required for their applications is not reasonable.

We tackle the problem of SSN application development for structural health monitoring by applying the design principles of *service-oriented architecture* (SOA). As a part of the Illinois Structural Health Monitoring Project (ISHMP) we have developed the Illinois SHM Services Toolkit, a suite of services implementing key SHM algorithms for modal analysis and damage detection, along with the middleware infrastructure necessary to provide high-quality sensor data and to transport it reliably across the sensor network. This software is open-source and available for public use at http://shm.cs.uiuc.edu/software.html. As these services are loosely coupled and dynamically composable, different SHM applications can be easily created. Because it can be augmented with services for other domains, the tool suite also provides a common, extensible platform for SSN application development. This paper provides an introduction to this tool suite.

2. SERVICE-ORIENTED ARCHITECTURE

With the exponential growth in available computing power over the last 50 years, the complexity of computer software has likewise increased dramatically. Advances in the fields of programming language design and software engineering allow application developers to deal with this complexity by dividing the software system into smaller, manageable parts. Notably, *object-oriented programming*, which encapsulates data together with the methods used to operate on it, and *component-based software architecture*, which proposes building applications as a composition of self-contained computing components, have been instrumental to the design and development of large-scale software systems. Expanding on this idea, *service-oriented architecture* (SOA) has recently been proposed as a way to bring this design philosophy to building dynamic, heterogeneous distributed applications spanning the Internet (Singh and Huhns 2005, Tsai 2005).

Services, in SOA terminology, are self-describing software components in an open distributed system. The description of a service, called a *contract*, lists its inputs and outputs, explains the provided functionality, and describes non-functional aspects of execution (timeliness, resource consumption, cost, *etc.*). Data is passed among the services in a common format. An application consists of a composition of a number of linked services within a middleware runtime system that provides communication and coordination among them. Unlike traditional component-based architectures, services do not have to be tightly coupled with each other or operate on the same computer; indeed, the services do not have to be explicitly linked to each other until execution time. Services do not need to know who provides the input data they require, or from where it comes. Different applications can be built from the same set of services depending on how they are linked and on the execution context (Gu et al 2005). This approach makes for dynamic, highly adaptive applications without the need to revisit and adapt the implementation of each service for a particular application context.

We see SOA design principles as being applicable in sensor network context as well as on the Internet. Smart sensor networks (SSN) often consist of numerous independent nodes, eac h an embedded computing platform

with a processor, memory, and a radio transmitter. As such, SSN applications are by definition distributed and thus require communication and coordination for parts of the application running on different nodes. SOA has been proposed to address the inherent problems in designing complex and dynamic SSN applications (Liu and Zhao 2005, Mechitov et al 2007). Building an application from a set of well-defined services moves much of the complexity associated with embedded distributed computing to the underlying middleware. This approach also fosters reuse and adaptability, as services for a given application domain can be employed by a multitude of applications.

Perhaps more importantly, SOA provides for a separation of concerns in application development. That is, application designers can focus on the high-level logic of their application, service programmers can concentrate on the implementation of the services in their application domain, and systems programmers can provide middleware services (reliable communication, time synchronization, data aggregation, *etc.*) that enable the services to interact. In sensor networks, which at this stage are principally used by scientists and engineers, the application designer is likely to be the user of the application as well. This situation makes it especially important for the high-level design of the application and the domain-specific algorithms used by the services to be separated from the low-level infrastructure necessary to make the system work. SOA in SSNs makes it possible to compose and deploy, on-the-fly, complex applications through a web-based user interface suitable for non-programmers (Ravazi et al 2007). User-driven SSN programming is a relatively young research area with few working implementations, but it holds the promise to lower the barriers to entry in sensor network application development and to make SSNs a more attractive solution.

3. SOA FOR SHM APPLICATIONS

The Illinois SHM Services Toolkit (http://shm.cs.uiuc.edu/software.html) provides an open-source software library of customizable services for, and examples of, SHM applications utilizing SSNs. The library is currently implemented on the Imote2 platform (Intel 2005) and allows SHM of structures in a distributed manner. It is based on the Stochastic Damage Locating Vector (SDLV) algorithm, which requires synchronized acceleration data from a network of sensors and uses the Natural Excitation Technique (NExT) and the Eigensystem Realization Algorithm (ERA) for system identification.

3.1 Wireless Sensor Platform

The wireless sensor platform used in this research is the Imote2 (see Fig. 1), which is well-suited to the demands of SHM applications. It has a low-power X-scale processor (PXA27x) with variable processing speed to optimize power consumption. It incorporates a ChipCon 2420 802.15.4 radio with an onboard antenna (Antenova Mica SMD). The onboard memory of the Imote2 is one of the features that sets it apart from other wireless sensor platforms and allows its use for the high-frequency sampling required for dynamic structural monitoring. It has 256 KB of integrated RAM, 32 MB of external SDRAM, and 32 MB of flash memory (Intel 2005).

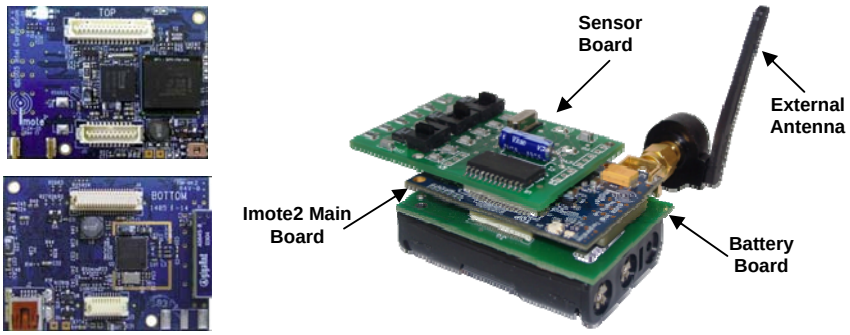

Figure 1. Top and bottom of Imote2 main board (left) and stackable configuration (right).

As with many wireless sensor platforms, the Imote2 employs TinyOS as its operating system. TinyOS is tailored to the specific constraints of sensor network applications. In particular, it occupies a small memory footprint while efficiently supporting complex programs. TinyOS applications are implemented NesC, a C-like programming language which supports the concurrency model of TinyOS (Levis et al 2005). While TinyOS has been adopted by many sensor network applications and has quite a large user-community, it can be a daunting undertaking for engineers lacking such specific programming experience to develop code for their applications.

3.2 SHM Application

The Illinois SHM Services Toolkit is based on the Distributed Computing Strategy (DCS; Gao and Spencer 2008). Nagayama and Spencer (2007) and Nagayama et al (2008) developed the necessary middleware services, compiled the numerical sub-functions, and created the complex application code to implement DCS on a network of Imote2s. The details of the DCS implementation on the Imote2 can be found in Nagayama and Spencer (2007), however for completeness it is briefly summarized here.

First, the sensor network is divided into overlapping clusters of sensors (Fig. 2), which monitor sections of the structure. One of the nodes in each cluster, or local sensor community, is assigned as the cluster head and handles most of communication and data processing within the community. In addition to tasks inside the community, the cluster head communicates with the cluster heads of neighboring communities. When signaled to do so, all nodes in the network simultaneously measure acceleration and synchronize the data.

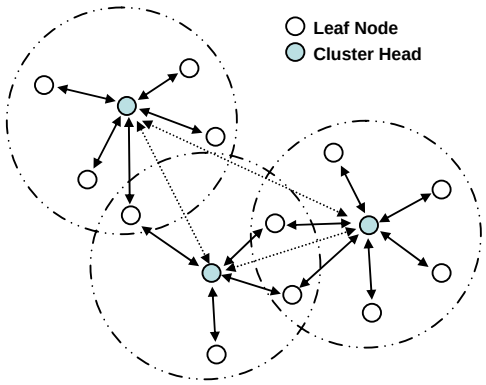

Figure 2. Network topology.

Upon the completion of synchronized sensing, the following tasks occur with in each local sensor community:

1. Acceleration data is shared to perform model-based data aggregation (Nagayama and Spencer 2007) using the Natural Excitation Technique (NExT; James et al 1993).
2. System identification is performed to determine the dynamic characteristics of the subcomponent of the structure using the Eigensystem Realization Algorithm (ERA; Juang and Pappa 1985).
3. The Stochastic Damage Locating Vector (SDLV; Bernal 2006, Nagayama and Spencer 2007) algorithm is applied to determine if the structure has sustained any damage and to determine the location of the damage.

Finally, the cluster head of each local sensor community compares its results with the cluster heads of adjacent (and overlapping) communities to ensure that the results are consistent. If consensus between the cluster heads is achieved, only the outcome of the damage detection method is forwarded to the base station. This entire process is illustrated in the simplified flowchart shown in Fig. 3.

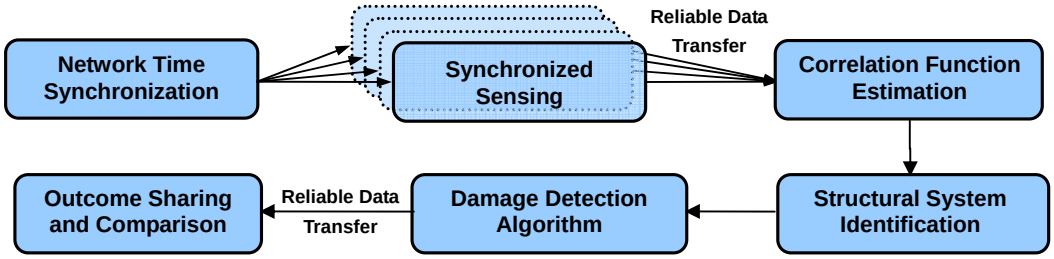

Figure 3. Flowchart of DCS implemented on a network of Imote2s.

This implementation has been experimentally verified on a laboratory-scale three-dimensional truss (Nagayama and Spencer 2007). While the application proved to be successful in both network functionality and the ability to detect damage, it is not written in a modular way, thus lacking the flexibility necessary for adaptation to different applications. The following section describes the SOA that has been created from these middleware services and application code to provide the necessary modularity and flexibility.

3.3 SOA for SHM Application

The contents of the Illinois SHM Services Toolkit can be divided into three primary categories: (1) foundation services, (2) numerical services, and (3) tools and utilities. In addition, a library of supporting numerical functions that are common to many SHM algorithms is provided; including fast Fourier transform (FFT), singular value decomposition, Eigenvalue analysis, etc.

3.3.1 Foundation Services

In SOA terminology, services are high level, self-describing building blocks for distributed computing applications. The foundation services implement functionality needed to support the application and other services. They include gathering synchronized sensor data, reliably communicating both commands and long data records, and providing accurate and precise timestamps to collected data. These services in combination can be used by applications to achieve synchronized sensing from a network of sensors.

The *Unified Sensing* service acts as a wrapper around the standard TinyOS sensing interface for the Imote2, extending its functionality to include timestamping and providing transparent support for a variety of sensor boards. Data for all sensor channels, together with a single set of associated timestamps, is returned to the application in a single, shared data structure. A very compact data representation format is used, which encapsulates all information necessary to recreate the sensor values, yet is memory-efficient. This common data format is appropriate for passing directly to the application services described below. This complete and self-contained data is easy to pass around and modify without hard-coding connections between components that use only parts of this data (Rice et al 2008).

A *Time Synchronization* service provides consistent, network-wide global timestamps for sensor data, making it possible to meaningfully compare data collected from multiple sensors. The *Resampling* service then takes the globally-timestamped sensor data and resamples it to a specified fixed sampling rate. This resampling is accomplished in a memory-efficient way with a resampling filter that is applied to the data one block of at a time.

Since sensor data loss is intrinsic to wireless systems and undermines the ability to perform system identification and detect damage (Nagayama et al 2007b), a *Reliable Communication* service, which eliminates data loss, is needed for sending commands and data between sensor nodes. The *ReliableComm* service employs four distinct reliable communication protocols, chosen automatically based on the type of communication, to eliminate data loss in an efficient manner.

3.3.2 Numerical Services

These services provide the numerical algorithms necessary to implement the SDLV method on the Imote2s. These services may also be used independently in other application contexts. For each numerical service, an application module to test the algorithm on both the PC and the Imote2 is provided. The numerical services are as follows:

1. *CFE*: Returns the Correlation Function Estimate (CFE) via FFT calculation. CFE uses two synchronized discrete-time signal vectors to obtain their CFE with user-specified number of FFT points and spectral window. In the case of DCS, the output of CFE is used as the input to the ERA or SSI system identification services.
2. *ERA*: Performs the Eigensystem Realization Algorithm (ERA). This time-domain system identification service uses the impulse-response function, or in the case of the NExT algorithm (James et al 1993), the correlation functions, to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, modal participation factors, EMAC values and the matrices which define the state-space model of the structure: A, B and C).
3. *SSI*: Performs the covariance-driven Stochastic Subspace Identification (SSI) algorithm. This time-domain system identification method uses the cross correlation functions to determine the modal characteristics of the structure (damped natural frequencies, damping ratios, mode shapes, and the matrices which define the state-space model of the structure: A and C)
4. *SDLV*: Performs output-only, model-based damage detection using the Stochastic Damage Locating Vector (SDLV) method. The inputs of SDLV are the modal characteristics determined by one of the system identification service. More detailed information on this method can be found in Nagayama and Spencer (2007).

Documentation is provided for each service and test application, giving more detail on requirements and formats of the inputs and outputs for the service.

3.3.3 Tools and Utilities

The application tools are complete applications that facilitate common tasks throughout the design, testing, deployment, and monitoring of the SHM applications, while utilities offer a set of basic testing and debugging commands to be included with existing applications. Included are utilities for resetting nodes remotely, listing the nodes within communication range of the local node, and changing the radio channel and power for local and remote nodes.

The application tools include, among others:

- *LocalSensing:* This tool allows sensor data to be collected while a single Imote2 is connected directly to the PC (i.e., no radio communication is required). It allows developers to test the functionality of sensor boards and develop driver software for new boards.
- *RemoteSyncSensing:* A network-wide distributed application, this tool is used to collect synchronized data from multiple sensors. It synchronizes the network prior to sensing, collects timestamped data and then resamples the data locally to account for any jitter or non-uniform delay in the start of sensing for each node (Nagayama and Spencer 2007). All data and commands in *RemoteSyncSensing* are sent between nodes using the *ReliableComm* service, eliminating data loss.
- *TestRadio*: Tests the bidirectional communication between a sender node and a group of receiving nodes, and output the packet loss rate (in each direction, and round-trip).
- *imote2comm*: A basic terminal program for interfacing with the Imote2 through the Imote2 Interface Board's USB port. It uses the serial port UART interface to open a telnet-like connection with the mote.

3.4 Broader Applicability

While the toolkit was developed to implement DCS using the SDLV method on a network of Imote2s, its service-oriented architecture lends itself to further expansion and in the development of SSN applications for SHM. As a simple example, the figure below represents how the system identification method can be swapped out in an SHM application. In keeping with the SOA framework, these interchangeable services share the same input and output parameters.

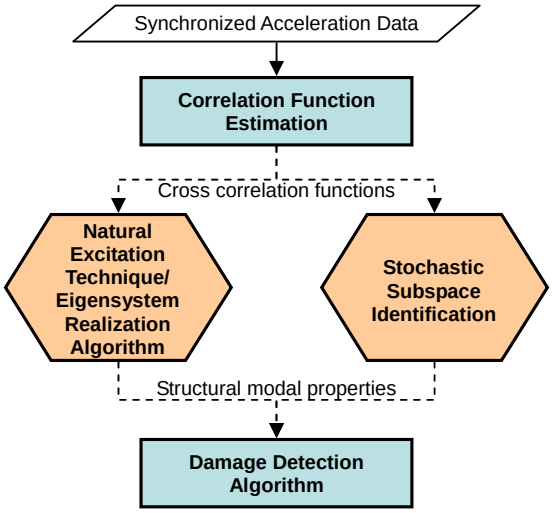

Figure 4. Alternate services for SHM application development.

In its current form, the Illinois SHM Services Toolkit requires application programmers to simply provide the code necessary to interconnect the services and tools in a way that makes sense for their applications. In general terms, this code serves the following functions:

- Run service *X* at node **A**
- Send a control message to node **B** to run service *Y*
- Send results from **B** to **A**
- Run service *Z* taking as inputs the outputs of services *X* and *Y*

The Toolkit gives examples of how to connect the services to create applications. Future work will automate this process to use "drag-and-drop" graphical programming.

4. CONCLUSIONS

This paper describes the Illinois SHM Services Toolkit that is open-source and available for use. The toolkit of services was developed using the design principles of service-oriented architecture (SOA). This SOA-based approach creates a framework which allows application programmers to more easily create applications for SHM systems deployed on SSNs. As a result, the framework allows more researchers, and ultimately application engineers, to design and implement successful SHM systems without the requirement of how the underlying middleware and numerical services are implemented. Both the services provided in the toolkit and the approach in general are transferrable to scenarios beyond SDLV-based SHM.

REFERENCES

Bernal, D. (2006). Flexibility-Based Damage Localization from Stochastic Realization Results," *J. of Engineering Mechanics* **132:6**, 651-658.

Farrar, C.R. and Doebling, S.W. (1997). Vibration-based health monitoring and model refinement of civil engineering structures, *Proc. 1st International Architectural Surety Conference.*

Gao, Y. and Spencer Jr., B.F. (2008). Structural health monitoring strategies for smart sensor networks, *NSEL Report, Series 011*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/8802.

Gu, T., Pung, H.K. and Zhang, D.Q. (2005) A service-oriented middleware for building context-aware services. *J. Network and Computer Applications* **28:1**, 1-18.

Intel Corporation Research (2005), Intel Mote2 Overview, Version 3.0, Santa Clara, CA.

James, G. H., Carne, T. G., & Lauffer, J. P. (1993). The natural excitation technique for modal parameter extraction from operating wind turbine, Report No. SAND92-1666, UC-261, Sandia National Laboratories.

Juang, J.-N. and Pappa, R. S. (1985). An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics* **8:5**, 620-627.

Kim S., Pakzad S., Culler D., Demmel J., Fenves G., Glaser S. and Turon, M. (2007). Health monitoring of civil infrastructures using wireless sensor networks. *Proc. 6th International Conference on Information Processing in Sensor Networks*, 254-263.

Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2005), TinyOS: An Operating System for Sensor Networks. Ambient Intelligence, Weber, W., Rabaey, J.M., Aarts, E., Eds. 115-148, Springer, Berlin, Heidelberg.

Liu, J. and Zhao, F. (2005). Towards semantic services for sensor-rich information systems. *Proc. International Workshop on Broadband Advanced Sensor Networks*.

Lynch, J. P. and Loh, K. (2006). A summary review of wireless sensors and sensor networks for structural health monitoring. *Shock and Vibration Digest*, **38:2**, 91-128.

Mechitov, K., Razavi, R., and Agha, G. (2007). Architecture design principles to support adaptive service orchestration in WSN applications. *ACM SIGBED Review*, **4:3**.

Nagayama, T. and Spencer Jr., B.F., (2007). Structural health monitoring using smart sensors, *NSEL Report, Series 001*, University of Illinois at Urbana-Champaign, http://hdl.handle.net/2142/3521.

Nagayama, T., Spencer Jr., B.F., Rice, J.A. and Agha, G. (2007) Smart Sensing Technology: A New Paradigm for Structural Health Monitoring, *Proc., 39th Joint Meeting of the US-Japan Joint Panel on Wind and Seismic Effects, UJNR.*.

Nagayama, T., Sim, S-H., Miyamori, Y., and Spencer Jr., B.F. (2007) Issues in structural health monitoring employing smart sensors, *Smart Structures and Systems*, **3:3**, 299-320.

Nagayama, T., Spencer Jr., B.F., Mechitov, K., Agha, G. (2008). Middleware services for structural health monitoring using smart sensors, *Smart Structures and Systems*, in press.

Razavi, R., Mechitov, K., Agha, G. and Perrot, J.-F. (2007). Ambiance: A Mobile Agent Platform for End-User Programmable Ambient Systems. *Advances in Ambient Intelligence, Frontiers in Artificial Intelligence and Applications* **164**, 81-106.

Rice, J.A., Spencer Jr., B.F., Mechitov, K. and Agha, G. (2008) Flexible Smart Sensing Framework for Structural Health Monitoring, *Proc. US-Korea Workshop on Smart Structures Technology.*

Singh, M.P. and Huhns, M.N. (2005). Service-Oriented Computing: Semantics, Processes, Agents, John Wiley and Sons, New Jersey.

Spencer Jr., B.F., Ruiz-Sandoval, M and Kurata, N. (2004), Smart Sensing Technology: Opportunities and Challenges, *Structural Control and Health Monitoring* **11**, 349–368.

Tsai, W.T. (2005). Service-Oriented System Engineering: A New Paradigm. *Proc. IEEE International Workshop on Service-Oriented Systems Engineering*, 3-8.