Design and Simulation of a SOA-based System of Systems for Automation in the Residential Sector

Navjot Kaur*, C.Stuart McLeod*, Atul Jain*, Robert Harrison*, Bilal Ahmad*, Armando Walter Colombo[#], Jerker Delsing[±]

*Loughborough University, UK; [#]University of Applied Sciences, Emden/Leer, Germany
[±]Lulea University of Technology, Sweden
Email: n.kaur@lboro.ac.uk, c.s.mcleod@lboro.ac.uk, a.jain@lboro.ac.uk, r.harrison@lboro.ac.uk, b.ahmad@lboro.ac.uk
awcolombo@et-inf.fho-emden.de, jerker.delsing@ltu.se

Abstract— **Today's process industries typically utilise a very large set of multi-disciplinary, connected, heterogeneous systems composed of a large number of components, from individual sensors to whole control, monitoring and supervisory control systems. The use of such process control systems spans a wide range of application domains from traditional process industries to power distribution and residential applications. Regardless of the application, the design, implementation, optimised operation, maintenance and monitoring of such complex systems is often difficult to effectively achieve. Therefore, there is a need for the migration of such systems to scalable and modular plug and play platforms supported by effective engineering tools.**

This paper outlines a distributed modular service-based approach to the implementation of a real-time residential automation system application composed of heating, home automation and vehicle parking systems. The main focus of this paper is on the design and simulation of this System of Systems (SoS) focusing on the home parking, heating and vehicle monitoring application components and their interactions. The use of a component-based engineering approach is reported on providing 3D modelling and simulation capabilities prior to the physical implementation of the system, and from which the orchestration logic can be automatically generated. A potential toolkit for SOA-based application design and deployment is suggested.

Keywords— Application Engineering, Component-Based (CB) Design, Distributed Systems, Process Monitoring and Control, Service-Oriented Architecture (SOA), System of Systems (SoS).

I. INTRODUCTION

Today industries are striving to engineer new business and control application architectures, which not only provide flexible solutions but also allow seamless interaction between control and business systems. Process industry control systems are becoming ever more complex, encompassing differing functionalities within the enterprise from environmental monitoring to real time control. At the same time energy costs are rapidly increasing driving the need to optimise energy consumption in the process domain. Thus, application systems are required that can interact more effectively with wider systems, such as energy management and environmental standards monitoring. The agility of process systems needs to be increased, whilst at the same time maintaining operational performance and reducing costs. Functionally, there is a need for better interoperability and integration of control functions on different hierarchical levels ranging from field level to various higher level applications, such as process control and operations management services. The potential solution proposed in this paper is to realise a modular reconfigurable system based on a platform of reusable distributed components integrated within a Service-Oriented Architecture (SOA). These components are distributed entities interacting through their data, event and service ports. The SOA-based approach potentially simplifies the integration of monitoring and control systems at the application layer.

II. SYSTEM OF SYSTEMS: CHALLENGES

The term System of Systems (SoS) refers to large-scale integrated and distributed systems that are heterogeneous and independently operable on their own, but are networked together for a common goal [5]. In brief, a successful system of systems is likely to be loosely coupled but tightly integrated, which is a desired characteristic of a distributed SOA system. SOA not only provide an excellent platform for developing systems from sets of services offered by process controllers, data acquisition systems, and legacy systems that have been exposed through services, they also offer the ability to extend system behaviour in a system of systems approach. The systems that constitute a SoS typically have the characteristics such as able to operate independently, managerial independence, emergent behaviour (i.e., SoS behaviours can't be anticipated from the behaviour of any one system), evolutionary development and graphical distribution. In order to develop a system of systems, tools are required to model the overall system requirements and architecture, model the communication between the systems, configure the systems, simulate and visualize the systems behaviour and interaction.

Interoperability between different systems is very important. Dynamic interoperability means the systems are able to comprehend, where necessary, the state changes that occur in other systems over time, and are able to take advantage of those changes. SOA provides interoperability in a way that the interoperating systems will be aware of services exposed by other systems. These services are discoverable and provide data definitions and metadata regarding its context, defined by WSDL. In order to create a system of systems, information from sub-systems may be required through some

defined interface or mediator. In order to promote the dynamic reuse of systems and components, the interfaces must be described with contextual information to allow correct understanding and integration into higher-level systems.

III. THE ADVENT OF SOA-BASED AUTOMATION

In recent years, the utilisation of service-based architectures, typically based on the use of Ethernet, in control systems has become an emerging approach to reduce the technological barriers between the low-level automation and enterprise systems. The importance of distributed systems has proven itself in the context of modern automation concepts. In addition to the horizontal data and control flow, the vertical data exchange especially for monitoring and control of process control applications is advantageous. These systems provide the potential to enable the design of complex but flexible applications that consist of several distributed objects or services, executed on different communicating nodes. Distributed systems, especially ones based on the SOA design principle, have been investigated in research project SOCRADES in both the discrete- and process-control domains [3]. One of the design goals is to establish a complete information chain from field level up to ERP-level based on these SOA technologies. It has been successfully demonstrated that it is possible to apply SOA at device level, using the Devices Profile for Web Services (DPWS) solution, standardized by OASIS [8]. The SODA [11] and SOCRADES and SIRENA [4] collaborative research projects have implemented and demonstrated the use of SOA in several application domains in manufacturing automation systems. The FP7 AESOP [9] project of which the research reported on in this paper is a part, is specifically addressing various process control requirements while using SOAs. This project aims to focus on the real-time requirements as one of the technical investigation topics, with the objective to demonstrate the feasibility of adapting cross-layer SOA in control and monitoring of processes [10].

IV. A MODULAR COMPONENT BASED DESIGN APPROACH

To meet the challenges mentioned above in section I, a distributed automation system is required in which various control components are interconnected with each other using service-based interfaces. The design and realisation of such a system requires: a) a suitable architecture, b) control devices with I/O modules for sensors and actuators, and c) engineering tools for designing and implementing service based architectures. To design a distributed control system using the Component Based (CB) approach, the architecture of the desired system can typically be categorised in a hierarchical structure composed of system, subsystem and components. A major advantage of such a modular system design approach is the utilisation of generic hardware and control software components that could be reconfigured to form new process configurations to suit different application variants [2, 14]. The modular component-based architecture enables efficient machine build and re-use of designs in order to optimise the system's lifecycle across many phases of use and with many supply chain partners.

The system engineering of distributed embedded systems requires the modelling and support of the units of distributed functionality. The system is conceived as a composition of interacting components, which are then mapped to real-time tasks. Ad-hoc specifications and design currently severely limit component reusability and therefore it is highly desirable to develop a formal framework that will allow for a systematic specification of reconfigurable components, which will be reusable by definition. In the CB design approach, the component functionalities are represented by the component's state, behaviour (i.e. control logic/state transitions), and operations and error information, which are encapsulated in each control unit. To provide flexibility and reconfigurability, user applications are composed by a higher level engineering tool, the Process Definition Editor (PDE) tool developed on the COMPAG and BDA projects ([2], [13,14]), which is not only responsible for defining component functionality and interfacing, but also the way components are mapped onto real-time tasks.

This can be observed that distributed applications require components, function units such as sensors, controllers, actuators, operator stations etc. and in SOA based design approach, these function units must interact transparently with one another via appropriate services to choreograph or orchestrate activities appropriately to suit the application need. An engineering tool suitable to define these components, their behaviour and run-time execution in a real-time scenario is proposed in this paper.

V. SOA BASED DESIGN FRAMEWORK

The use of SOA based technologies with the modular CB approach can provide an open, non-vendor specific approach to control systems, as well as enabling seamless enterprise integration. To design and implement modular CB systems based on SOAs, a step-by-step approach is discussed in this section.

A. System and Logic Definition

In distributed CB-based design, the whole system is divided into number of software components depending upon the application logic/process sequence. The software part of a system is basically an executable unit which can be deployed and composed at runtime. It provides the service perspective as well as the functional modularity aspects of a component. The software part consists of logical software objects, called control components, to represent and implement the logical control behaviour of each component. The behaviour of each control component is typically governed by a distributed finite state machine; comprising a finite number of states and their associated state transitional conditions [6].

B. Component Interaction

Component interaction is another important issue that has to be resolved when specifying system configuration. There are number of interaction methods which are widely used in process-based and object-based systems: client-server, producer-consumer(s) and publisher-subscriber. The client-server communication model can be considered as a distributed application that partition tasks or workloads between the

providers of a resource or service, called servers and service requesters, called clients. It is used in industrial computer systems; however, it has limitations such as the blocking nature of communication and additional unknown delay due to central server. Therefore, the producer-consumer model is often considered better suited for real-time systems because it is non-blocking and supports one-to-many interactions. Publisher-subscriber interaction combines the useful features of the former two methods, and is widely used in practice. Web-Services offer a particular set of component interaction capabilities, which need to be appropriately configured by the engineering system to best effect in the context of application requirements.

The main aim of employing SOAs on embedded devices down on the factory floor is to enable cross-layer and cross-system interactions. The commonly used protocol today for employing SOAs is Simple Object Access Protocol (SOAP), using the verbose XML language and more recently OPC-UA. XML has excellent support today from a large number of software vendors, which makes it an open and standardised way in exchanging data between devices from different manufacturers using different operating systems and application. One benefit from the SOA approach is that message parsers can be automatically generated for each message class. This reduces the need to manually write software for serialisation of messages.

C. *Simulation*

System simulation is advantageous both at design time and during the system operational phase. Simulation capabilities (e.g., to simulate the time-dependant, dynamic system behaviours) have to provide strong support for testing various aspects of the system design in a virtual form prior to the final system implementation in order to minimise design errors and compress design time. Simulation capabilities need to be provided for, and adapted to, each engineering application that the system design involves (e.g. covering the control, process, and mechanical system aspects). However, these domain dependant simulation capabilities have to be integrated in a form that will enable multi-disciplinary engineering teams to assess the level of completion and quality of their specific design with regard to the characteristics and behaviours exhibited by the final (virtual) system.

During the system's operational phase, simulation capabilities are required in order to commission off-line operational changes in a virtual form, i.e., so that scenario-based analysis and testing can be conducted without altering the normal operation of the real system. Such off-line simulation capabilities can be used for various purposes, from specific engineering assessment of system changes, re-design or re-configuration to projection of system load, and also monitoring and maintenance, e.g., to replay system activity in a virtual form prior to failure.

The implementation of specific simulation capabilities and their integration into a global system simulation application tool is made difficult by the complexity and scale of today's automation systems that make it difficult to integrate a number of different hardware and software technologies. In addition, technological solutions implemented in order to increase a system's "non-functional" capabilities such as flexibility; design agility (e.g. re-configurability and re-usability) and maintainability need to be taken in account during the design and implementation of simulation related functions. By using simulation tools prior to physical system deployment the commissioning time of any given automated process control system can be reduced considerably.

In a similar fashion (to the previously mentioned capabilities of system simulation) is the possibility to create hybrid systems that consist of a mix of real- and virtual-components. This allows the evaluation of different possible scenarios, general system behaviour and overall system performance in a very flexible manner.

D. *Behaviour of SOA-Based Systems*

In SOA based system design, the main aim is creation of services based upon functionality. For control applications this will naturally map onto a physical device breakdown of the system, which can be composed of the "components" of the application. The building of a complete process control system requires the integration of these components as well as sequencing or linking of defined services to achieve the system goals. For the purpose of this document, the concept of orchestration and choreography are described as:

Orchestration is the arrangement and stimulation of components, to achieve the desired system goals. With orchestration the overall process remains coordinated from the perspective of one of the involved parties [7]. For automation systems the orchestrator remains responsible for the correct stimulation or monitoring of the other components during the system operation.

Choreography is the definition and sequencing of the steps required to achieve the desired collaborative system goals. This requires that each component shall be aware of its operation with respect to the other components in the control system and collaboratively operate in order to fulfil the system goals.

For large-scale control systems choreography offers greater scalability with inherently decentralised, distributed control. Management components are also required to provide error reporting, manage the modes of operation (e.g., such as automatic and manual modes of use) and monitor the "health" of the systems based upon the health of its underlying components. Orchestrated systems are currently the most widely used, as their behaviour is easier to visualise and understand, they are therefore often easier to maintain using current tools. However, as systems related research advances, and the development of tools to organise their behaviour matures, choreographed systems may become more prevalent.

VI. SYSTEM ENGINEERING TOOLS

To build a distributed application using SOA technologies the core set of tools are required for device development, system configuration, commissioning and maintenance. To assist the development of the application, system modelling

tools are required to extract data from shop floor systems and present it in a usable format for business systems.

As already discussed in Section III, a solution with the potential to increase flexibility and adaptability in automation systems is to use a reconfigurable CBD approach. A CBD approach had already been researched at the Manufacturing Systems Integration (MSI) Research Institute of Loughborough University. This body of research aimed to investigate the potential of a CBD approach in improving the flexibility and re-configurability in manufacturing automation [3] and [13, 14]. The key outcomes of the research studies at MSI institute are: an engineering visualisation environment implemented using the Virtual Reality Modelling Language (VRML), reusable control software embedded within automation components and a Process Definition Environment (PDE) in which the automation components can be configured, simulated and deployed.

In the PDE toolkit, the system is broken down into components categorised as actuators, virtual, process, sensors and non-control components. An actuator contains the information of geometry, kinematic and logic, a sensor contains information of geometry and state, virtual components have state behaviour but no geometry or physical movement while a non-control only has geometry information. Process components are very similar to virtual, but are used to describe the Geometric information for a component is represented using the VRML data format. The logic information of a component is described as a state transition diagram (STD). By representing the components using STD interaction between them can be achieved at the process level in a structured, readily understood and open way.

A system is built by combining the components geometry and behaviour. The geometric model is defined by linking the geometry using predefined link points associated with a component in a hierarchical manner so that the movement of a parent object is followed by all of it children. Interlocks are added to the components to stop unwanted behaviour during operation, process logic is added to this behaviour for automatic operation.

Once correct application behaviour in the system can be verified in the simulation, the application logic can be exported to XML. The XML file contains all the information required to configure and run the system. As highlighted in Figure 5, the XML file is used by the orchestration engine to dictate the operation of the SOA devices.

VII. Application Case Study: Residential Context/Process Automation System

A system of systems design approach for monitoring and control based on SOAs is discussed in this section. The use-case is a residential process control and monitoring system involving a district heating system and home parking system, which is virtually simulated and evaluated prior to its actual implementation. The application logic is designed to optimise energy usage in a range of weather conditions. For simplicity, a single family house with one garage and number of parking spaces is considered here. A car can be parked inside or outside, depending on the weather conditions. If a car is parked inside the garage, the district heating system is informed about the additional heat source, so as to optimise the energy consumption. If the car is parked outside in very cold weather, its minimum temperature is electrically maintained to protect the engine. Fig. 1 shows the concept of the system to be virtually modelled in the PDE tools.

Fig. 2 and 3 represent the various domains, service definitions and workflow scenario for the whole system.

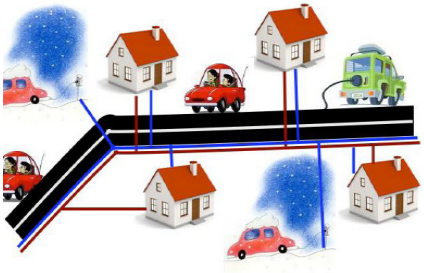

Figure 1. A residential car parking and district heating system proposed as a proof of concept for SoS approach [9].

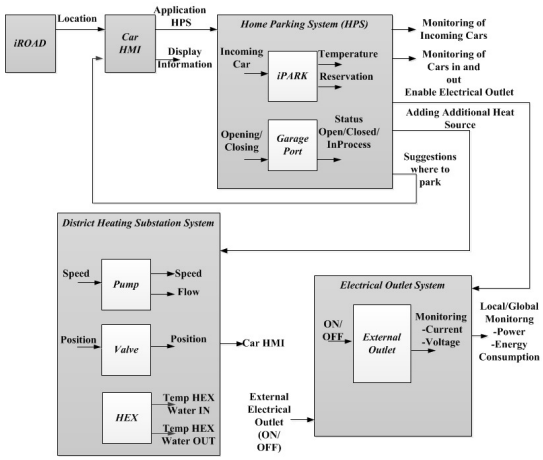

Figure 2. SoS approach and service definitions for each domain.

For each component, a 3D VRML model is designed geometrically in a component library. Based on the logic and behaviour of each component an STD is defined in each case. For example, the garage port will have state transition diagram as shown in Fig. 5. The STD's are defined for all the components and these state diagrams are interlocked to reach the desired process sequence defined in Fig. 3. For simplicity, a flowchart is provided here representing the behaviour of a single house, however, for the complete system of systems a large number of these behavioural models are interlinked with each other.

The component-based design of the system is outlined in Fig. 2. For example the ***Home Parking System (HPS)*** is composed of two subsystems: ***iPARK*** and ***Garage Port***. These subsystems are further divided into components and for simplicity of the concept; ***Garage Port*** is discussed in detail here. There is a possibility of parking a car inside or outside,

depending upon the outside temperature. For example, the car will be parked into the garage only if the garage is unoccupied and the temperature is very low. A *Garage_Status* sensor will be *(UNOCCUPIED/OCCUPIED)* depending upon the status of the garage. A *Temperature* sensor is used to indicate the outside temperature. This implies that if the *Temperature* sensor is *VERY LOW* and the *Garage_Status* sensor is *UNOCCUPIED*, the incoming car can be parked inside the garage. When the car reaches near the garage, a sensor named *Car_Detection* sensor will change its state from *OFF* to *ON*. The *Garage_Port* sensor is further interlocked with *Car_Detection* sensor i.e. the garage door will start opening when the *Car_Detection* sensor is *ON*. The component state changes are shown in Fig. 4 when the scenario is modelled, simulated and visualised in the PDE tools. The sensor state change is indicated by change of colour from *RED* to *GREEN*. In the PDE tools, the behaviour of components is defined in the form of STDs. The sensors are bi-state components and actuators can be two or four state, e.g., garage door is a four state actuator with two static and two dynamic states. Interlocking of various components is defined whenever there is a change from a *static (e.g., CLOSED)* to a *dynamic (e.g., OPENING)* state. A range of different simulated scenarios can be created by the user by altering the sequence-checks and interlocking between different components.

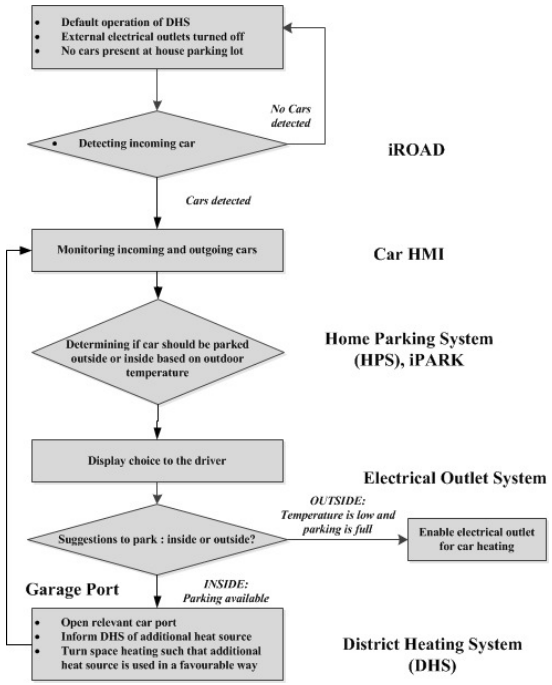

Figure 3. Proposed application scenario implementing a System of systems approach.

The distributed components and their interactions are modelled via this scenario based simulation. The system can be easily reconfigured. Component STDs can be reused where common functionality is identified across various components. Each system is composed of a number of service definitions embedded within it. For example, the service associated with the component *Garage_Port* is the status of the port, either setting it to *OPEN/CLOSE* or to get its status *(OPEN/CLOSE/INPROCESS)*. The PDE tools also facilitate the mapping of the developed system to a SOA-based runtime framework. The runtime orchestrator will receive the status of each component and sends control requests to the control components based upon the current state of the system and the application scenario definitions downloaded from the PDE design tools. Fig. 5 shows the overall architecture of the application in PDE tools and the capability to integrate this with the other systems using service orchestration.

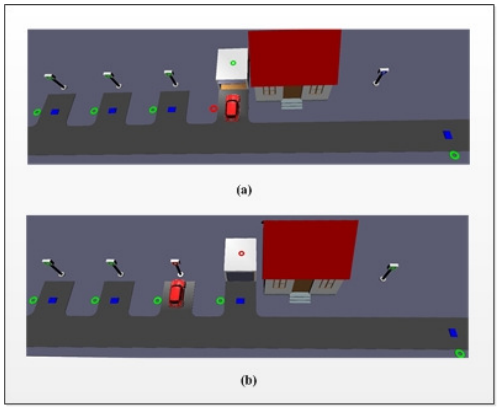

Figure 4. 2D view of HPS visualised and simulated in PDE tools: *Car_Detection* Sensor (a) *ON* (B) *OFF*, *Garage_Sensor* (a) *UNOCCUPIED* (b) *OCCUPIED Garage_Port* (a) *OPENING* (b) *CLOSED*.

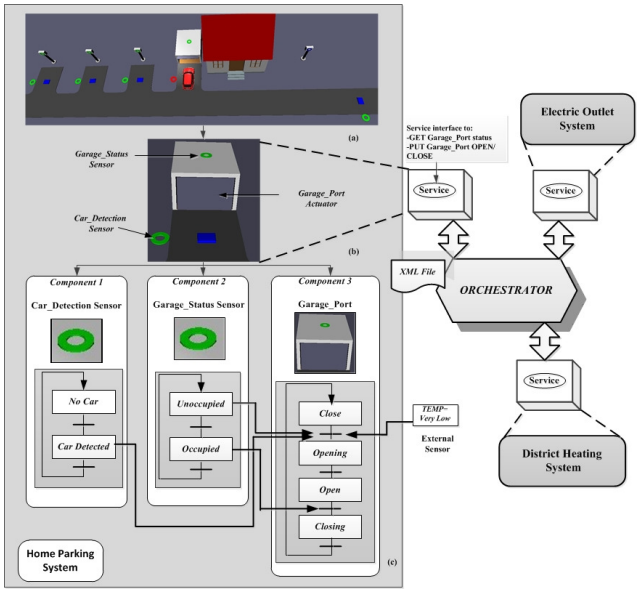

Figure 5. SOA based framework of SoS application using CB design approach (a) System- *Home Parking System* (b) Subsystem- *Garage_Port* (c) Components- *Car_Detection* sensor, *Garage_Status* sensor and *Garage_Port* STDs & interlocking.

To implement the system of systems concept, the proposed approach for a home parking system for a single house can be extended to number of houses interacting with each other through sharing and awareness of inter-system states via SOA. Figure 6 provides a schematic of the proposed system of systems approach for an automated residential parking system. The emergent behaviour of the overall system is very important. The PDE tools allow modelling of the overall system requirements and architecture, modelling of the communication between the systems, their configuration, simulation and visualisation.

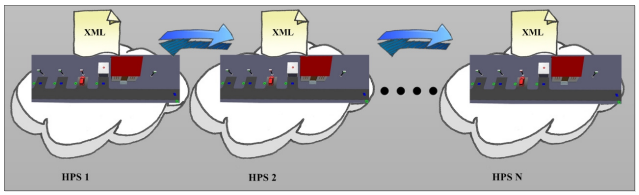

Figure 6. System of systems approach for Home Parking System

VIII. CONCLUSIONS AND FUTURE WORK

There is a lack of effective tools for the lifecycle support of SOA-based systems. The engineering tools described in this paper can help to facilitate the design, visualisation and simulation of a SOA based system of systems for monitoring and control of process-related applications. The stages in the development of a residential automation application using the PDE component-based design toolset have been outlined. The potential of the method has been shown in visualising the behaviour, and verifying the control logic for a residential automation application. The approach is readily extendable to include multiple loosely-coupled systems. The toolset enables the 3D visualisation and evaluation of different possible scenarios, general system behaviour and overall system performance in a very flexible manner. Such system simulation is advantageous both at design time and during the system's operational phase. Simulation capabilities (e.g., to simulate the time-dependant, dynamic system behaviours) provide strong support for testing various aspects of the system design in a virtual form prior to the final system implementation in order to minimise design errors and compress design time. The simulation scenarios are editable by alterating the interlocking sequences defined by the user, which further facilitates the dynamic deployment of the system. A key aspect of the on-going AESOP project is the design of SOA based SoS via new forms of application engineering tools. This is, illustrated in this research using the PDE tools to design such a system via a modular CB-based design process. Systems can thus be visualised and simulated in a graphical manner prior to their actual deployment. The system design method used further facilitates the design of service-based interfaces for every component, so that the services/functions can be orchestrated to integrate the control-level components with enterprise/business systems.

The future work related to the proposed application will include verification of system behaviour using Petri-net based tools. For service oriented design, an orchestrator based system interaction approach has been described in this paper; however, a choreographic approach will also be investigated in future, where components will interact in a truly distributed peer-to-peer manner. The future research work will include study of how such interaction between components is best deployed from the design tools within a SOA utilising Web Services.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the European Commission and the other partners involved in the FP7 project AESOP (www.imc-aesop.eu), and also the EPSRC for their support in the development of the PDE toolset via the BDA (Business Driven Automation) project.

REFERENCES

[1] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann. "Towards architecture for service-oriented process monitoring and control". In Proc. of the 36th Annual Conference of the IEEE Industrial Electronics Society (IECON-2010), Phoenix, *AZ.*, USA, Nov. 2010.

[2] R.Harrison, S.M.Lee & A.A.West, "Lifecycle Engineering of Modular Automation Machines". Proceedings of *the* 2nd IEEE International Conference on Industrial Informatics (INDIN '04*)*, Berlin, June 2004.

[3] S.M.Lee, R.Harrison, A.West, "A Component-based Distributed Control System, "In 2nd International Conference on Industrial Informatics (INDIN'04), June 24-26 2004, Berlin, Germany.

[4] S.Karnaouskos, P.Spiess, L.M.Souza, O.Baecker, A.Mensch, G.Starke, R.Monfared & M.Thron, "D6.1-Services Integration Concept for Field Related Data into Business Processes", SOCRADES project internal document.

[5] H.Bohn, A.Bobek & F.Golatowski, "SIRENA – Service Infrastructure for Real-time Embedded Networked Devices: A service-oriented framework for different domains", International Conference on Networking, Systems, Mobile Communications and Learning Technologies (ICN), IEEE Computer Society, 2006.

[6] T.Andreas, D.Saikou, T.Charles, "Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability and Composability for System-of-Systems Engineering", Journal of Systemics, Cybernetics and Informatics, 2007.

[7] D. A. Vera, A.West, R. Harrison, "Innovative Virtual Prototyping Environment For Reconfigurable Manufacturing System Engineering", Proceedings of *the* Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 223, Issue 6, 2009.

[8] A.W.Colombo, F.Jammes, H.Smit, R.Harrison, J.L.M.Lastra & I.M.Delamer, "Service-oriented Architectures for Collaborative Automation", In Proceedings of 31st Annual Conference of IEEE (IECON 2005).

[9] OASIS. Web Services Discovery and Web Services Device Profile (WS-DD). http://www.oasis-open.org/committees/ws-dd.

[10] AESOP-Architecture for Service-Oriented Process Monitoring and Control. http://www.imc-aesop.eu.

[11] J.Delsing, J.Eliasson, R.Kyusakov, A.W.Colombo, F.Jammes, J.Nessaether, S.Karnouskos and C.Diedrich, "A Migration Approach towards a SOA-based Next Generation Process Control and Monitoring," In Proc. of Annual Conference of the IEEE Industrial Electronics Society (IECON-2011).

[12] SODA-ITEA Consortium 2007,Software component DPWS C stack.. http://www.soda-itea.org.

[13] M.B Raza, R. Harrison, T. Kirkham "Knowledge based Services for Devices in Automation". KEOD 2011 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Paris, France, Oct 2011.

[14] R. Harrison, R.P. Monfared, L. Lee, "Bussiness Driven Automation for Powertrain Industry". Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on, Sept. 2009.