ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

TOWARDS A TIME-CONSTRAINED SERVICE-ORIENTED ARCHITECTURE FOR AUTOMATION AND CONTROL IN LARGE-SCALE DYNAMIC SYSTEMS

Gang Chen, Baoran An, Institute of Computer Application, Mianyang, P. R. China

Abstract

Rapidly changing demands for interoperability among heterogeneous systems leads to a paradigm shift from pre-defined control strategies to dynamic customization within many automation systems, e.g., large-scale scientific facilities. However, today's mass systems are of a very static nature. Fully changing the control process requires a high amount of expensive manual efforts and is quite error prone. Hence, flexibility will become a key factor in the future control systems. The adoption of web services and Service-Oriented Architecture (SOA) can provide the requested capability of flexibility. Since the adaptation of SOAs to automation systems has to face time-constrained requirements, particular attention should be paid to real-time web services for deterministic behaviour. This paper proposes a novel framework for the integration of a Time-Constrained SOA (TcSOA) into mass automation systems. Our design enables service encapsulation in filed level and evaluates how real time technologies can be synthesized with web services to enable deterministic performance.

INTRODUCTION

Today, the automation systems face challenges due to the changing demands on interoperability from their users. Formally, mass and standardized output was a key factor for the competitiveness. Nowadays and in the future the customization will become more and more important. Furthermore, the development cycles of these systems are about to become much shorter. This leads to a paradigm shift from mass production to mass customization within the dynamic system [1][2]. This means that the systems must be rapidly designed, able to convert quickly to the new models and able to integrate technology.

Currently, the engineering process for automation is characterized by high time-consuming manual configuration efforts. For example, in device-centric facilities for basic scientific research, all device properties and the data to be transferred must be defined by an automation engineer. Any modification of such a system requires an at least partial manual reconfiguration. Most of the time, the flexibility of current systems is limited to the pre-defined boundaries of the system. Therefore, the manual engineering is a major obstacle on the way to future fully automated systems.

Furthermore, the advance of engineering technologies relates closely to information technologies (ITs). Since design and operation of a dynamic system needs numerous types of decision-making at all of its hierarchy levels of automation, prompt and effective decisions not only depend on advanced reasoning techniques, but also on real-time data gathering and processing [3]. Every major development of automation has been supported by the advancement of IT. For example, the widely adoption of computer numerical control (CNC) made flexible manufacturing systems (FMSs) feasible; the technologies for distributed control systems (DCSs) made the interconnection of various parts in large automation systems practical. That is the reason why more and more cases in scientific scenarios rely on the professional provides of IT software solutions to replace or advance their conventional systems.

By today, the software infrastructure of big systems is often organized in vertical automation layers accordingly, shown as figure 1.

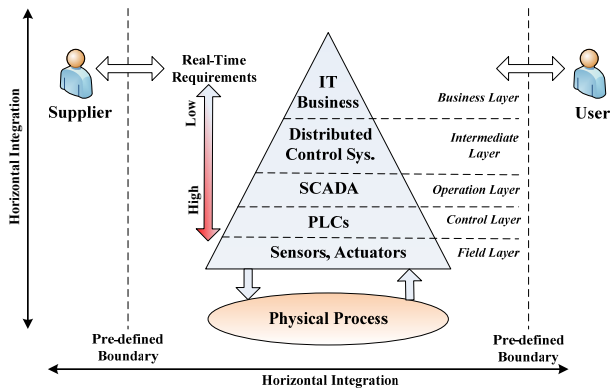

Figure 1: Automation Layers.

At the field layer, sensors and actuators provide interface to control physical processes. At the control layer, Programming Logic Controllers (PLCs) read from sensors, process them and generate new signals for the actuators. This process is repeated cyclically and autonomously on the basis of pre-defined control logic. At the operation layer, the data of the sub-system is gathered and visualized by industrial control software suite, like Supervisory Control and Data Acquisition (SCADA). And then, all the sub-systems are interconnected by distributed control systems based on Ethernet. This layer acts as a mediator between operation layer and business layer. It is in charge of collecting and integrating data from all operations for upper layer, and translating control orders into concrete control commands. Finally, the business layer contains software functionally for planning purpose. From the communication aspect, the layers differ in their requirements on real-time communication, especially for device-centric dynamic systems.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

We are motivated to investigate the standard device-centric Service-Oriented Architecture (SOA) and try to find a possible way to fulfil interoperability requirements of large systems. This paper has two main contributions: (1) we present a web service communication policy for non-standard application requirements; (2) A Time-Constrained Service-Oriented Software Architecture (TcSOA) is outlined that supports the convergence of heterogeneous automation layers. This work represents a partial aspect of the overall objective of reducing the engineering effort.

DEVICE-CENTRIC SOA IN LARGE SCALE SYSTEMS

The reduction of engineering effort is crucial in many research projects. This section presents our proposal to use web services as a communication backbone throughout a system. For this purpose, the SOA in automation systems are introduced briefly.

SOAs are software design patterns stemming from the IT field. The first group trying to use SOA in control systems propose that new services expose only their own certain functionality, and legacy applications have to divide their functionality into components and then wrap each component by a web service. That means, the implementation of each service is not visible from outside. The services embedded in the same system are loosely coupled: They operate independently from each other, their interactions are stateless, asynchronous and not context-related [4][5].

In device-centric automation systems, the services represent the functionalities of individual devices or processing modules, like simulating cells. The behaviour of the overall system is controlled by the coordination of all services. Due to independency and interoperation of services, the device-centric systems can be arranged by sequence-controlled services, which imply that reengineering effort for modifying existing automation structures can be minimized. Whereas the components of current big systems are interconnected and tightly-integrated, the automation engineers have to deal with any tiny modification with much manual effort, even in the initial setup stage of a system [6]. Furthermore, such type of composition must be guaranteed by an orchestration engine, which provides particular strategies to composite services, schedule the controlled services and to move up the newly composed service sequence to a higher layer.

Unfortunately, apart from the aforementioned advantages, the device-centric SOA approach is not adequate for describing the real-time constraints in many cases, due to non-real-time communication policies. Many techniques and standards are used for realizing device-centric SOA. For example, the web services are used as communication protocols that are called by SOAP RPC [7] through SOAP engines, like Apache Axis 2 [8], and WSDL [9] is used as service call interface.

As demonstrated in previous work, future dynamic systems can adopt and benefit from SOA, especially in some device-centric scenarios. Furthermore, this framework can be applied in horizontally and vertically within the automation systems. The architecture is illustrated in Figure 2. The services provided by device producers are orchestrated though a particular middleware or directly used as service modules in dynamic systems.

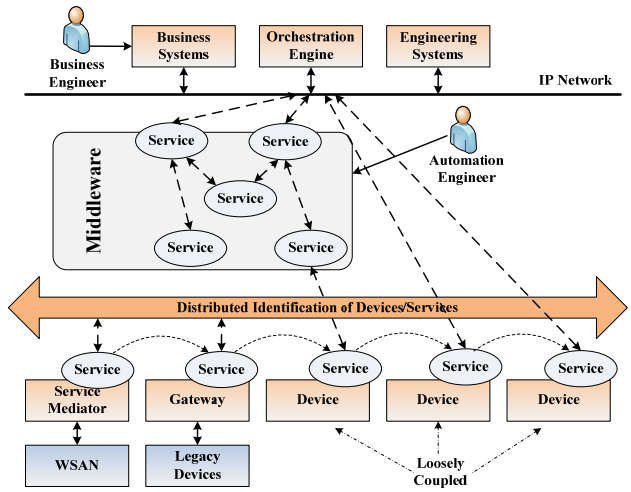

Figure 2: Device-Centric Service Framework.

DEFINING A WEB SERVICE COMMUNICATION POLICY

Actually, web services are not really suitable for implementing applications with non-standard communication requirements. This section discusses the real-time communication in SOAs again.

Real-Time Views in SOA

In time-constrained SOA, service providers can enable real-time services, i.e. invocations of services must be completed within specific timing constraints. As well, in device-centric SOA environment, the device producers can provide services that follow time-constraint requirements. Besides that, all operations in systems that are facilitated by SOA have to abide by time-constraints.

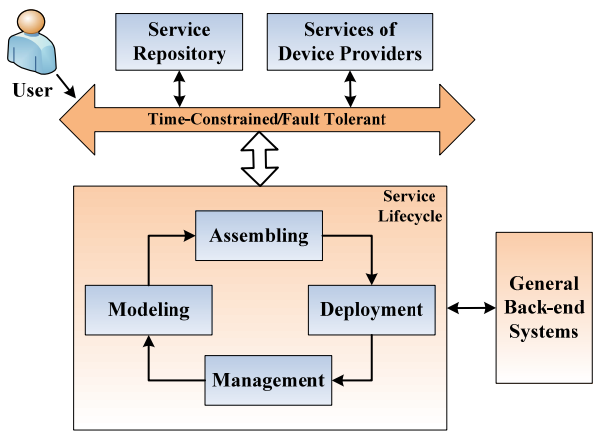

Figure 3: Lifecycle of SOAs.

To clarify the time-constraint requirements in SOAs, we will now look at the lifecycle of a SOA [10][11], including modelling, assembling, deployment and manage-

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

ment, as shown in Figure 3. It is necessary to investigate the common techniques in service lifecycle of traditional SOAs in terms of the needs of real-time requirements.

In the modelling phase, individual services are initialized and specified. Many attributes and properties are setup, including the functionalities and time-constraints. The sequenced execution order is also formed and shaped in this phase. Service providers have to restrict timing-related attributes to some particular values, i.e. the response time, invocation frequency and service timeliness. After that, individual services can be orchestrated into top-level services through the assembling phase. During orchestration, the consistency and completeness of adjacent services should be checked and evaluated. The control logic in orchestration engine is responsible for service composition. All these operations should have bound response time. In deployment and management phases, in order to guarantee time-constraints, the usage of resources should be reserved, like network bandwidth, power supplies, CPU time and build-in hardware modules.

Basic Idea: Real-Time Communication Policy

Real-time communication policies have been discussed in the scientific community for more than a decade and despite their advantages, however, they are still not established in enterprise level applications. Here we pay attention to two types of time-constrained scenarios requiring for real-time communication policies.

The first scenario is to examine the communication between individual services. Since services might exist in different remote nodes, the real-time communication is a critical issue that needs to resolve to achieve the objectives of TcSOA. We have analysed different setups with service communication to identify which parts could be omitted to reduce time requirements. Basically, messaging techniques are the most important parts of service communication implementations. Firstly, messages to be exchanged between services need to be serialized in real-time for marshalling/unmarshalling[11]; The resource requirements, such as channel bandwidth and CPU time, need to be reserved. Because the messages may cross over multiple layers and corresponding protocols, a service can provide various levels of messages. Regardless of which level, a time-constrained message should possess the following characteristics:

- Limited response time set, like the minimum and maximum response time
- Recovery strategies (such as message retransmission mechanisms)
- Maximum data capacity
- Degree of concurrency, like the maximum number of receivers that the message is sending to
- Cost and required resources

The quality of messaging on real-time properties almost determine the real-time achievements of the whole services, which implies that the service providers have to adjust their services according to the application scenarios, such as resource constraints, data exchange, modelling methods and real-time requirements.

The second scenario is to investigate the real-time communication policies in device-centric environment. Due to the development of fully integrated and flexible end-to-end SOA platforms, i.e. Enterprise Service Bus (ESB), the SOA architecture is made practical in real-world applications [12]. However, SOA-steered applications are still not adequate for device-centric dynamic systems, because there are so many hardware modules existing in this type of systems, like sensors, actuators and other PLC-based devices.

The main devices used in the bottom layer of a control system are front-end computers and PLC controlled devices, such as industrial computers, step motors and servo motors. In traditional dynamic systems, the devices newly integrated are connected to some PLC applications and the PLCs are linked to a front-end computer via real-time Ethernet, as shown in Figure 4.

As most devices, like sensors and actuators, are off-the-shelf products, it is unrealistic to require all device producers to provide web services for the devices accordingly, especially with real-time constraints. Meanwhile, we have noticed that real-time constraints are only required by small groups of devices in the real-world control systems, like some particular sensors and their corresponding actuators. Thus, it is vital to identify the parts with and without real-time requirements in device-centric control systems.

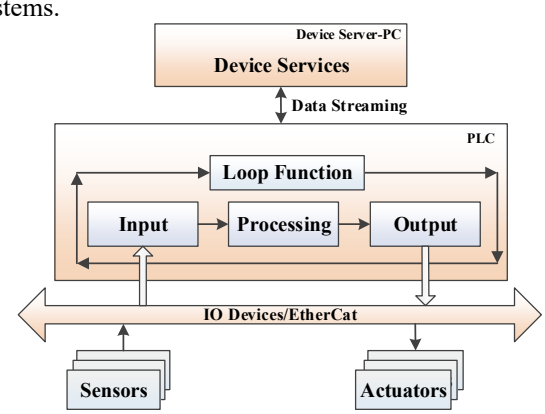

Figure 4: Current Control System.

Here, we propose a novel concept Device-Cluster-Virtual-Module (DCVM) to instead of the concept of individual device. A DCVM is a small group of physical tightly-coupled devices, which can provide some functionality to the outside world. Within a DCVM, the devices are interconnected via real-time communication based industrial data bus, like field bus. That means, inside the DCVM, all devices are tightly-coupled while DCVMs are loosely-coupled between each other. In order to control the behaviours of the devices and data processing, an embedded orchestration engine is integrated in each DCVM, which provide control logic by automation engineers for device communication and management. For the purpose of connecting with outside SOA and hiding the PLC details from outside, we should establish a small embedded web service engine for each DCVM, which will reorganize the PLC applications to permit automatic

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

identification of functional entities. A sample DCVM is shown in Figure 5.

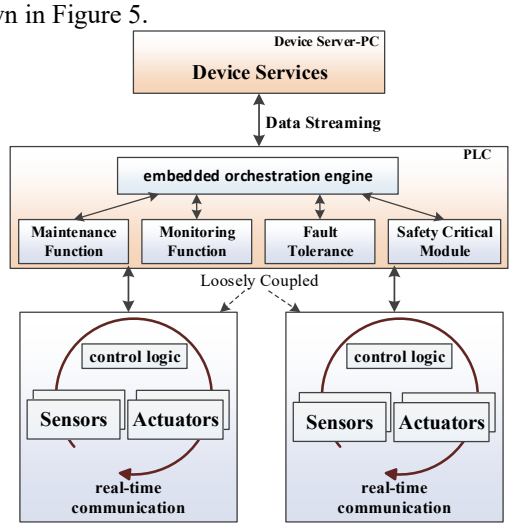

Figure 5: A schema of DCVM.

This design pattern allows PLCs to offer standard web services and users can use standard tools to access the bottom layer of automation systems. The technical details of PLC will be invisible from users. Thus, automation engineers can operate their familiar development environment to develop and deploy control applications.

SYSTEM ARCHITECTURE

Real-Time Requirements

The architecture envisioned in this paper aims to device-centric automation systems involving devices and subsystems that are very different in nature and typical time-scale of operations. Considering our preceding statements and real-world requirements, we outline the following non-general classification of time-constraints:

- The business layer transforms business logic and basically has no QoS requirements and need more supports in best-effort way.
- Communications throughout the whole system, which typically are of less critical real-time requirements, and whose reaction times need to reside within several hundreds of milliseconds (i.e., less than 200ms).
- Communications between SOA services belonging to the same categories, i.e., data services, middleware services, are always constrained to hundreds of milliseconds, i.e., 100ms.
- Communications among devices located in the subsystems with the same boundaries. Their time-constraints are typically in the period of hundreds of milliseconds.
- Interactions at the same module (DCVM), i.e. the couple of sensor and actuator, usually require 10ms-level time-constraints.
- Interactions at different module but belonging to the same mechatronic unit are also required to perform in 10ms.

The goal of the architecture is to let services with different time-constraints coexist in the same system. Solutions, like optimized scheduling algorithms, dedicated hardware and software, efficient protocols and resource reservations can be of benefit to achieve the goal.

Software Modelling and Real-Time Properties

Software modelling is a basic issue in almost every application. As aforementioned, a service-oriented model has been chosen as the software paradigm, since it increases design and execution flexibility. A service is characterized as an elementary unit which enable the building of distributed applications in a decoupled way, i.e. services located at remote nodes can communicate with each other via messages or events. Hence, a service can be considered as a data channel, as shown in Figure 6. It receives some input data, processes them and produces results that can be delivered to other services or transferred to the higher layers.

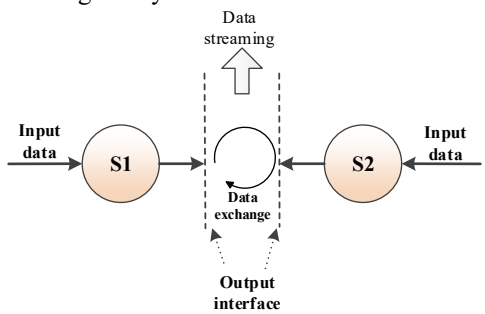

Figure 6: Data Channel Model for services.

Since services are encapsulated as functional units, it is possible to extend the applications via interconnecting the services. Aided by distributed techniques, it looks like that these services reside in the same pool. Actually, service-oriented applications are groups of services in the form of graphs [13], as illustrated in Figure 7. The ellipses at the top of the picture denote decoupling points (messages), where individual components can be connected and disconnected without any change of the connected systems. Within a reuse component, the circles denote services, and the connecting lines or arrows are the messages exchanged between them. In this model, service connections can have dependencies that can be both functional (data transferring and portal computability) and non-functional (time-constraints and resource-constraints). In this context, the real-time properties have to be guaranteed whenever there is a functional or non-functional variation. In any case, it implies a transition from one state (Sinit) to a target state (Starget). Time-constraints are going to achieve whenever the transition from Sinit to Starget is performed in less than the specified time td as:

$$f(S_{target}, S_{target}) < t_d$$

Fundamentally, the goal of real-time properties is that the function f must be time-bounded. Therefore, f depends on a number of levels that must be carefully considered, like hardware, middleware and operating systems[14].

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA152

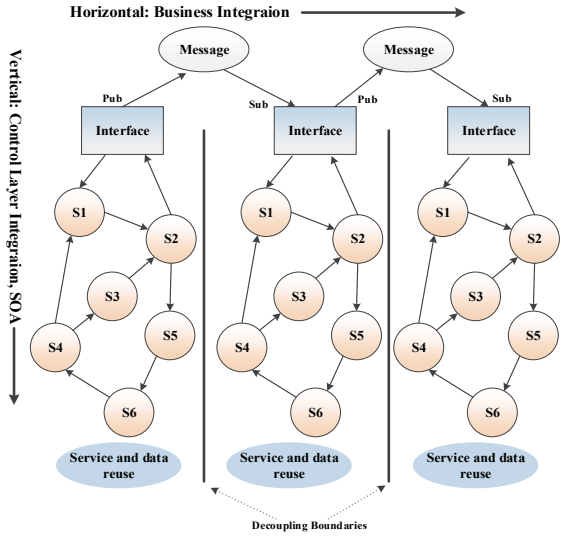

Figure 7: SOA Application as a Graph Structure.

Architecture

Figure 8 depicts the envisioned Time-Constrained SOA (TcSOA) development environment that integrates time-constrained web services and consists of four functional layers – application layer, execution layer, real-time service layer and hardware layer – which contain several components to meet the demands of web service based automation systems. The integration of devices and standard services will be enabled by encapsulating functions inside communication policies. Generally, in this envisioned architecture, what we are trying to do is that Everything-as-a-Service (XaaS) for automation.

As shown in Figure 8, the control and field layers are encapsulated in DVCMs. DVCMs cover functionalities of all physical devices. As well, the upper levels have been changed. They contain all infrastructural components of the TcSOA framework. Potential users of this framework are automation engineers who are domain experts but do not necessarily know web services.

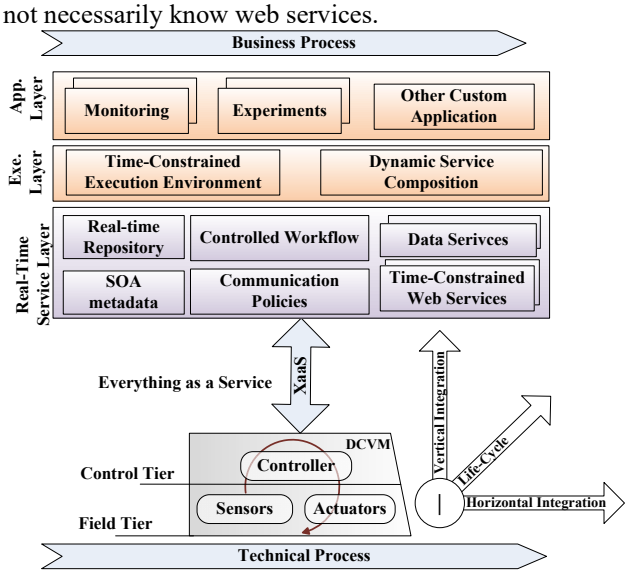

Figure 8: Architecture for Time-Constrained Services

CONCLUSIONS AND OUTLOOK

Future automation and control applications will need to be developed at a rapid pace in order to capture the required agility. Typical software development approaches need more manual efforts and posses less flexibilities. Thus, future control systems should be adjusted to new paradigm of distributed large systems with collaboration and multi-layer interactions. In this paper, solutions based on web services have been analyzed to find their potentials for deployment in large scale dynamic systems. This would be possible with migration of current automation functions to web services by considering the SOA requirements of applications. Significant work need to be invested towards further investigating the interdependencies and needs of all targeted service domains as well as the technologies for realizing them.

REFERENCES

[1] L. Duerkop, H. Trsek, J. Otto and J. Jasperneite, “A Field Level Architecture for Reconfigurable Real-time Automation Systems”, in *Proc. WFCS’14*, Toulouse, France, May 2014.

[2] H. ElMaraghy and H. –P. Wiendahl, “Changeability- An Introduction”, *Changeable and Reconfigurable Manufacturing Systems and Transformable Factories’09* , pp. 1-24.

[4] I. M. Delamer and J. L. M. Lastra, “Loosely-Coupled Automation Systems Using Device-Level SOA”, in *Proc. INDIN’07*, Vienna, Austria, Jul. 2007, pp. 743-748.

[5] Lars Duerkp, Juergen Jasperneite and Alexander Fay, “An analysis of Real-time Ethernets with Regard to Their Automatic Configuration”, in *Proc. WFCS’15,* Palma de Mallorca, Spain, May 2015.

[6] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of IoT and Industry 4.0”, *Ind. Electron. Mag.*, vol. 11, no.1, 2017, pp. 17-27.

[7] M. Loskyll, J. Schlick, S. Hodek *et al*, “Semantic service discovery and orchestration for manufacturing processes”, in *Proc. ETFA’11*, Toulouse, France, Sep. 2011.

[8] M. Gudgin *et al*, “SOAP Version 1.2 Part 2: Adjuncts”, June 2003.[online] Available: `http://www.w3.org/TR/soap12-part2/.`

[9] Apache Axis 2, `http://ws.apache.org/axis2/`

[10] WSDL, `https://www.w3.org/TR/`

[11] R. High, S. Kinder, S. Graham, “IBM’s SOA Foundation: An Architecture Introduction and Overview”, 2005.11.

[12] W. T. Tsai, Y. –H, Lee and Z. Cao, “RTSOA: Real-time service-oriented architecture”, in *Proc. SOSE’06*, Shanghai, China, Oct. 2006, pp. 49-56.

[13] M. –T. Schmidt *et al*., “The Enterprise Service Bus: Making Service-Oriented Architecture Real”, IBM System Journal 44, no. 4, 2005, pp. 781-798.

[14] M. G. Valls, I. R. Lopez, I. F. Villar, “iLand: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems”, IEEE Trans. on Industrial Informatics, vol. 9(1), pp. 228-236, 2013.