2009 33rd Annual IEEE International Computer Software and Applications Conference

Software Methodologies for the Engineering of Service-Oriented Industrial Automation: The Continuum Project

J. Marco Mendes[1], Axel Bepperling[2], João Pinto[2],
Paulo Leitão[3], Francisco Restivo[1], Armando W. Colombo[2]
[1]*Faculty of Engineering - University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal (e-mail: {marco.mendes, fjr}@fe.up.pt)*
[2]*Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany (e-mail: {axel.bepperling, joao.pinto, armando.colombo}@ de.schneider-electric.com)*
[3]*Polytechnic Institute of Bragança, Quinta S[ta] Apolónia, Apartado 134, 5301-857 Bragança, Portugal (e-mail: pleitao@ipb.pt)*

Abstract

Service-orientation represents a new wave of features and solutions by bringing closer Information Technology to the industrial domain, particularly factory shop floors. The service-oriented automation software entities (designated here by bots) used in such approach requires a short set of methodologies and software targeting their specification for both computer systems and embedded automation devices. The present work explains the adopted methodologies and software developments for the engineering of service-based automation systems. The main contents focus on the specification of a framework for the development of bots and supporting engineering tools that are part of the Continuum project. The paper also does an overview over the engineering steps from the system design to the operation, and focuses the importance of the maintenance of automation bots. Such applications will contribute to decrease the development time and reduce the components' interdependency, offering enough flexibility for automatic reconfiguration of shop-floor layouts.

1. Introduction

Distributed computing systems are a natural evolution of isolated computing, not only for extending the limitations in terms of processing power and consequent drawbacks, but also to assimilate software to the real nature of things. Examples of dispersions and their relations with the environment can be found in natural systems (such as ecosystems [1]) and also in human-made ones, like theme parks. Whatever the extent is, the distribution should also be handled with some kind of arrangement and obey to imposed laws, else the behavior would be chaotic and non-sense.

Particularly in the scope of this work, distribution of equipment, operators, products and information can be found in modern industrial production systems that should be ready for both mass-production and mass-costumization. A large number of factors are critical in the effective operation of such flexible production lines, including the number of product options, manufacturing operation of each one, product type, workstation capacity, processing time of the operations at each station, material handling capacity at each work station, and overall material handling capacity [2]. Therefore, the resulting data to be processed, besides being enormous, may also be constantly in change.

Distributed software components are already being used in the form of distributed objects, function blocks and services. The last one, in form of *Service-oriented Architectures* (SoA) and *Service-oriented Computing* (SoC), is hitting right now the domain of industrial automation systems. The idea of "*service-oriented computing to provide a way to create a new architecture that reflects components' trends toward autonomy and heterogeneity*" [3] dominates the view of the future trend. Also its growing maturity in the business and e-commerce ground, are seen as step forwards for a *seamless integration* [4] of resources from different levels. Currently there are several efforts dealing with these subjects, such as the SOCRADES project (http://www.socrades.eu) to show the rising tendency in the research and industry community.

Generally, SoA does only view the system by its services, but less important are their providers and requesters. These entities may represent automation devices, software components and others, with the

0730-3157/09 $25.00 © 2009 IEEE
DOI 10.1109/COMPSAC.2009.66

special ability of encapsulating resources under the shell of services. They represent a new approach that is different from the traditional used Programmable Logic Controllers (PLC) in automation. These entities are sometimes called smart devices [5], other times possessing different names, but having in common representation as service enabled autonomous control entities, devices actuators and sensors [6]. This work adapts the word *bot* (as an abbreviation for the word "robot") to designate interacting software components and mediators of hardware devices, robots and other resources that may be found at the shop-floor.

Even if there are already existing tools for different engineering steps of service-oriented systems, little focus has been given to the nature of service-oriented entities besides their services and interaction mechanisms. Additionally, and considering the novelty of it in industrial automation, the adoption of SoA principles from the business level to the shop-floor has to be smooth or be adapted to face the new environment. In this context, the paper presents the developed Continuum project with the goal to provide an integrated engineering tool for service-oriented automation systems, especially the specification and management of the life-cycle of automation bots. The mechanisms of interaction and control patterns using service-orientation, and also a more advanced engineering procedure are out of scope and are not presented in this paper.

The outline is the following: section 2 discusses background information and requirements for the specification and development of software methodologies and supporting applications for service-oriented automation; section 3 steps into the architecture of the Continuum project, focusing more on the details of the bot framework and the development studio; section 4 presents an overview about several engineering steps since the system design to the execution using the described approach; and finally the paper resumes the conclusions and indicates future work.

2. Background and requirements

Most applications developed today rely on a given middleware platform which governs the interaction between components, the access to resources, etc. [7]. Service-oriented Computing is a new paradigm that evolves from the Object-Oriented Computing (OOC) and component based computing paradigms by splitting the developers into three independent but collaborative entities: the application builders (also called service requestors), the service brokers (or publishers), and the service developers (or providers) [8]. The scope of services applications goes beyond organizational boundaries, such as e-Business. So, integration and collaborations are key issues. And, dynamic provision and evolution of services are a new area of Software Engineering [9], that change the way that software is developed and deployed.

The initiative in service-based systems replies its echo in the concept of collaborative automation [11] in the sense of autonomous, reusable and loosely-coupled distributed components. Service computing and orientation is here viewed not only as a form of communication, but more a philosophy that software entities should adopt by sharing resources and representing their needs. As said before, this also stands for a new way of design and thinking for automation engineers, supported business managers, and software engineers that have to develop the necessary tools and methodologies. Figure 1 depictures this view based on the requirements flow from the automation and business "worlds" for the specification of the necessary software.

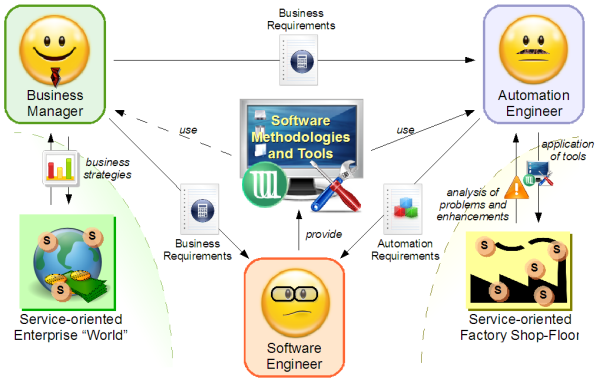

Figure 1. Requirements for software development of service-oriented automation.

For the success of a company, the marked must be correctly analyzed and business strategies have to be planned, which can maximize the usage of external suppliers and clients, and consequently produce the most favorable outcome (more details on this subject can be obtained from game theory, particularly the best response strategy and the Nash equilibrium). Of course, since the environment is not static, flexibility and adaptability are keywords of enormous importance. The same is valid for efficient planning and management of industrial shop-floors by the automation engineers. A perfect balance between mass-production and mass-customization is essential in the new markets of ever changing demands.

Even if not seen at the first sight, both worlds are service-enabled, and this does not always mean from

the technology point of view. In fact, the technology side of services is nowadays more and more close related to Web services. Therefore, software engineers have the task to provide the right tools to facilitate the life of both "communities" for service-oriented environments, from the concept to the technology. Business managers may used, for example, to get statistical information of production and consequently plan new strategies. But the main usage is for the automation "people" so that they can easily plan and configure factory cells.

There are numerous requirements for the development of such systems and tools that can be found in the requirement tables of several projects (see for example the SOCRADES and SODA projects and several related publications [11] [12] [13]). In this context it is important to discuss the requirements that software engineers have to attend to specify configurable software entities and engineering tools. Major requirements are:

- Service-oriented architectures as a reference model for the specification, operation and integration of automation systems;
- Maintain some compatibility to traditional standards, such as IEC 61131 and IEC 61499;
- Easy development environment for automation engineers and integration capabilities in business levels;
- Device considerations such as use of low-cost embedded devices with plug-and-play capabilities, energy efficient, performance restrictions, security, reliability and portability of code;
- Reuse, composition, aggregation, extension and simplification of services and software entities;
- High-level process description for component behavior and inter component relations, supported by handling of undocumented and exceptional events;
- Prepared for decentralization, autonomy of entities, automatic reconfiguration and/or simple manual reconfiguration.

The main challenge in the requirements and proposed features, such as autonomy and re-configurability are quite a problem when developing software for the engineering of those systems. In the end the software tools should provide the necessary easiness so that they can be uniformly used and, from the other side, have the necessary features.

3. The Continuum project: specification and development

Based on the requirements, it was decided that the basic building blocks that compose the distributed system should be configurable software components assuming different tasks, in a form of a component-based service-oriented framework. As referred in the introduction, the designation *bot* was adopted to identify a software component (moreover in section 3.1). To design, configure and maintain bots, there is a need of specific tools, that are user-friendly and speed-up the development, using a high-level programming approach (visual languages). Figure 2 represents a schematic diagram of the used design principle.

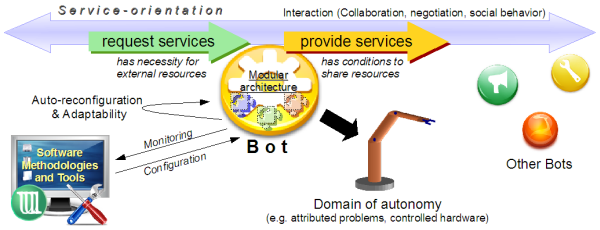

Figure 2. Design concept for the software of service-oriented industrial automation.

The project was baptized *Continuum Development Tools*, named after the continuum concept used in physics and philosophy. First developments were started by integrating already developed software components, in special the PndK (Petri nets development toolKit) [16], under the same umbrella. Along with the integration, it was identified that several software packages are needed, namely: a framework for developing bots, engineering tools for the design and managing of bots and several utilities (mainly libraries) for supporting activities (e.g. such as communication and interface for devices).

Figure 3 represent a component diagram with the several grouped software components that were planned for the initial compendium of the Continuum project. Target systems range from traditional PC's (especially for the engineering tools) to the devices that should embed the generated bot code. The groups are categorized by the automation bots, their supporting engineering tools and additional utilities (in form of libraries) to support the development. The main component would be the *Continuum Bot Framework* (CBF) for the development of bots and their functional modules, inspired in the anatomy of living beings. Another component, the *Continuum Development Studio* (CDS) that is based on an extensible Document/View framework, provides an engineering tool for service-oriented bots, for example, supporting the visual description, analysis and simulation of their behavior (for now, in Petri nets formalism according to the definition of T. Murata [15]). Both automation bots and engineering tools are explained in more detail

in the next subsections. The Utilities package includes several reused software libraries and tools, some developed internally others adopted from the outside, such as the SOA4D DPWS library (available at https://forge.soa4d.org) providing facilities for the development of Web services and the Qt toolkit (see http://qtsoftware.com), used mainly as a graphical toolkit for human interaction in the CDS.

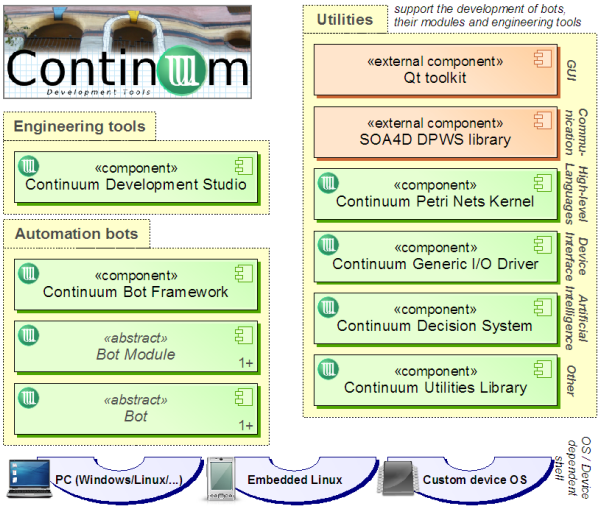

Figure 3. Main software components of the Continuum project.

The main languages of development are C and C++. The development environment was generated and is maintained using several tools. As such, Subversion (http://subversion.tigris.org) is used as versioning control system and CMake (http://www.cmake.org) was the choice for the building system (permitting cross-platform development and generation). Additionally, documentation is generated using Doxygen (http://www.doxygen.org/). There is no specific software project management tool that is used, since the group is constituted by few people and the development is normally done at the same place.

3.1. Automation bots and their framework

Bots are a common term used in computer games to designate non-human players in a virtual world that should behave as similar as possible as the ones made of flesh and bone. Therefore, methodologies of artificial intelligence and nowadays scripting are used to specify their behavior, whatever it is based on known information or on completely new input. Bots are also used in other fields with similar meaning and/or extending it also for the use of repetitive software tasks, such as chatbots, Web bots and tutor bots for e-learning (as industrial robots for physical tasks). In a general way, it is correct to say that a bot is the software part of a robot or simulated being.

The adoption of this designation in this work is easy to comprehend: from one side the question of representing someone or something, from the other side the connotation to robots that are widely used in industrial automation. Still open, is the internal organization of bots that is a special concern to the developers and to the end users. As such, a modular approach was adopted to specify several functional and reusable modules that compose in the end a full integrated bot. The reader may look to a module as an organ of a living being, providing specific functions and properties. For example, a bot that is mediator of an industrial robot may have a module for communication, a Petri nets interpreter, and also a device interface (so it may read/write signals from/to the robot device).

Bots that implement several functions require a consistent anatomy to deal with the different function modules ("organs") in order to fulfill the necessary requirements. Other problems may arise from the asynchronously operating modules, possible data inconsistencies and concurrent processes/threads. As a whole, the integration of modules into a full functional bot must be considered. Similar to what happen to most of the animals that have a nervous system, "impulses" or signals generated by modules should be routed correctly to the destiny and be interpreted. This can be considered as a form of loose integration, particularly event-based integration in which modules interact by announcing and responding to occurrences called events [16].

The main basis for the development of bots is the Continuum Bot Framework. A class diagram centered on the CBF and realizations of modules and bots is shown in Figure 4. A module can be defined by inherit the CBFAbstractModule class and adding special functionality to it. For example, the Petri Nets Kernel Module uses the functions and structures of the Continuum Petri Nets Kernel library. For the DPWS Module, the external SOA4D DPWS library was used to create a communication module, so that bots could use it to communicate to others, via exposition of services and consumption of others' services. An independent bot (integrated as a stand-alone application or library) can be obtained by deriving CBFAbstractBot, add some custom code and specially combining required modules. See the example of a Mechatronic Bot in Figure 4 that depends on several modules.

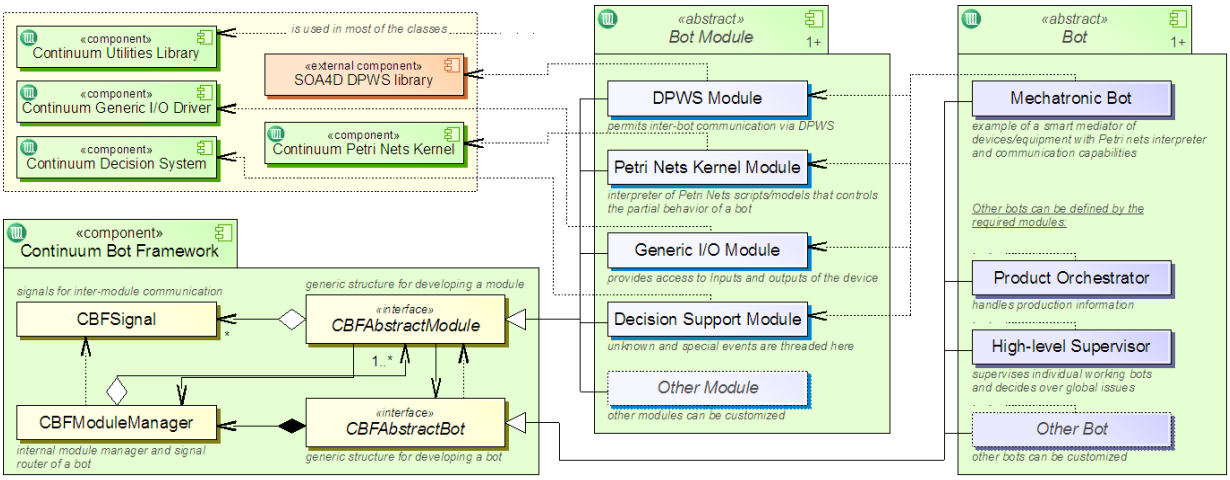

Figure 4. Class diagram and realizations of the Continuum Bot Framework.

Signals are used for intra-specific communication of a bot, i.e. event-based interaction between its modules. A signal is created from the CBFSignal class and several parameters and user-data can be set in the signal's instance. Signals are sent by a module and routed via the intermediate CBFModuleManager that has a reference to each module. The receiving of signals and their analysis is done asynchronously by each module. When a signal is received, it is saved in the local queue of the module. Internally, a module represents a threaded close-loop that analyses the local queue of received signals. Whenever a signal is popped out from the queue, a code corresponding to this event is executed. The used signals mechanism can be compared in the functional way to the Signal/Slot approach from the Qt toolkit [17].

3.2. Development studio for engineers

During the past decade, many kinds of distributed computing systems have been proposed and built and they also differ considerably in how they are programmed [18]. The used programming languages can vary from the conventional languages, to high-level forms of representation, dedicated to distributed/parallel systems. The initial choice for such a language was Petri nets (specified originally by C.A. Petri in his thesis [19]), not only because they assimilate some of the languages used in automation (see the IEC 61131-3 standard), but also due they wide spectrum of features and applicability, possibilities for extensions and solid mathematical foundation. In its core, a Petri net is an abstract formal model of information flow [20], which can be represented graphically in the form of a graph by specifying the relation of places and transitions.

The application of Petri nets can range typical systems with defined behavior to more complex ones with distributed participants. In any case, system engineering and associated tools are required to facilitate the developer's intervention. From the Petri nets side, the practical usage is limited by the lack of computer tools which would allow handling large and complex nets in a comfortable way [21]. Therefore, the Continuum Development Studio (CDS) is intended to provide a user-friendly environment for several engineering tasks of service-oriented automation systems, since the specification and configuration of automation bots, analysis and simulation, until the operation of the system. Figure 5 represents a screenshot of the CDS, simulating a Petri net control model.

The development was based on a port and natural evolution of the previous PndK, enriched with a multi document/view type framework (similar to the model-view-controller architectural pattern) and additional tools. The framework was created on an insufficient basis of the used Qt toolkit (that has in fact the support for model-view programming in form of classes, but does not provide a framework for their management and integration into an application). Basically the framework includes a document manager class for supervising documents and their views, a project explorer to aggregate documents in a logical way and the abstract classes from which the developer can create customized documents and views. The document manager permits the creating of document and view instances in the fashion of the factory method pattern and also the customization of their tools, e.g. menus, tool bars and other widgets. File handling (via the operations of new, open, save, etc.) is also handled in an integrated manner for all types of documents. For

now, Petri net and text document types (and corresponding views) were implemented.

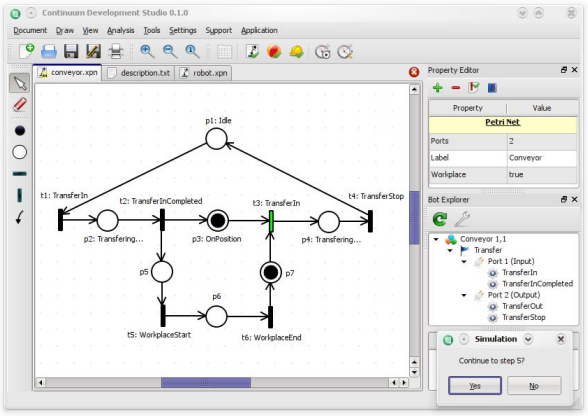

Figure 5. Continuum Development Studio showing a simulated Petri net.

Petri net document/view permits the design, analysis and discrete simulation of Petri nets. Additionally, a customized Property system was developed to allow the enrichment of Petri nets and their elements with information that can be used, e.g. to associate the Petri net model to the behavior of an automation bot. There is a built-in orchestration engine that is able to coordinate and synchronize services (using the SOA4D DPWS library) according to the workflow described by a Petri net. Configuration of bots (that include the Petri Nets Kernel Module) is done mainly by describing their expected behavior via a Petri net model, including the request of external services, exposition of its own services and device access.

4. The Continuum project: system engineering overview

Once the software is fully completed to be used, the question now is how to use it for specifying the automation system. The following section describes the several required engineering steps since the system design until its operation and reconfiguration.

First of all, the used case study scenario compromises a flexible production system with two work stations (that can be used by operators and robots), several conveyors that route production pallets into/out of the system and to the workstations, and also two lifters that make the interface between the upper and lower levels of conveyors. The system exists physically (Figure 6.a) and was also 3D modeled in DELMIA (Digital Enterprise Lean Manufacturing Interactive Application), used for simulation, monitoring and to provide the connection of virtual devices (see Figure 6.b). Prototype devices were connected to the several equipment units (conveyors and lifters) for hosting the builded automation bots (initially un-configured). The used tools and engineering methodology were applied to this scenario with the goal of transferring pallets to the workstations and introduce some flexibility in the design and maintenance of the system.

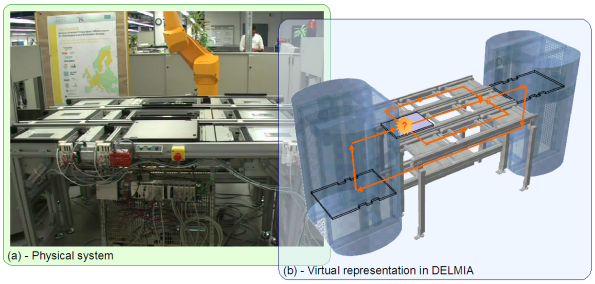

Figure 6. Physical and virtual representation of the case study scenario.

At the current time, the design phase compromises the use of the virtual representation in DELMIA mainly to export the connection information in XML format to the CDS. The CDS is employed for designing and analyzing the Petri nets template models for describing the behavior of the bots (Figure 7.a). When importing the device/connection information from DELMIA, several steps are done by the CDS: 1) instantiate Petri nets models for each bot based on the designed template models, 2) create the necessary properties of the Petri nets models so that several parameters of the given information from DELMIA are written on the models (e.g. bot/device information, connections, ...), and 3) based on the "enrichment" of the Petri nets models, composition of models can be done for creating connection logic and for the overall system analysis (Figure 7.b).

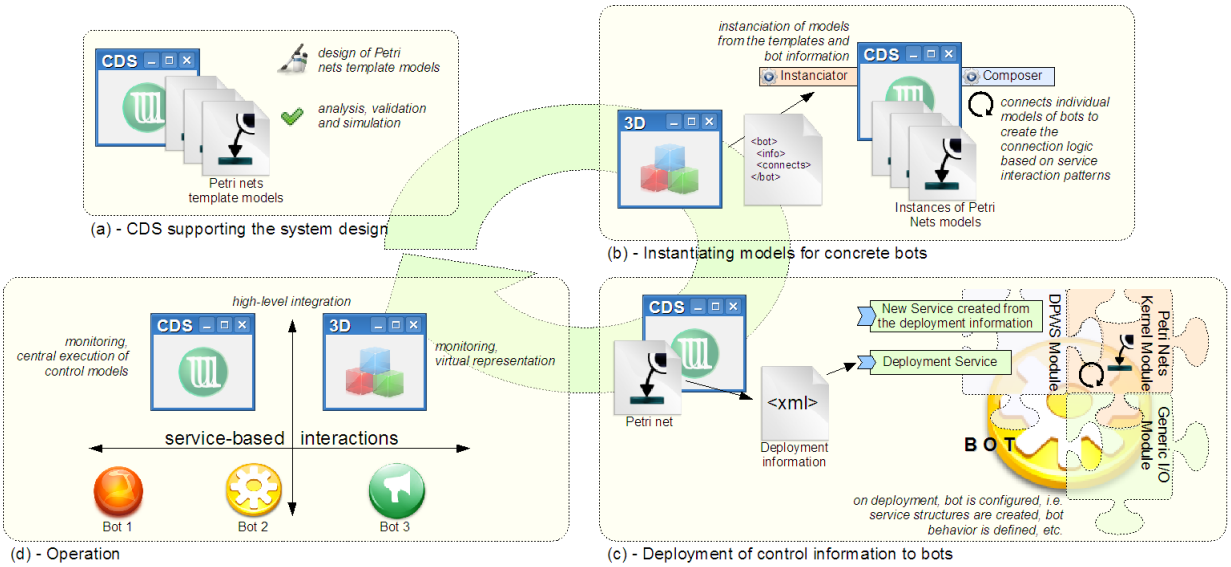

Figure 7. Engineering steps using the Continuum approach

After the analysis and simulation (that can be done with the CDS and also with DELMIA, providing services of the virtual devices), bots must now be configured. The process of deploying a service that encapsulates its logic as a Petri nets model to a bot that provides an embedded Petri Nets Kernel Module is depicted in Figure 7.c. The deployment functionality is a standard feature of the DPWS and is exposed as a dynamic deployment service. The target and the deployment service can be discovered by stacks built-in discovery service. After deployment a new service endpoint has been added and the execution of the services logic has been initiated. Deployment information includes the Petri nets behavior model, connection information of neighbors (required services), provided services by the bot and also extra configuration information for the other modules of the bot. The bot will configure itself (and its modules) and is then ready for operation.

Operation means autonomous behavior of bots according to their defined model, plus internal exception handling and the exposition and requesting of services by the different bots and other software components that are on the system (Figure 7.d). Higher level features in the service approach includes also the aggregation of services into one (simplifying the outside view), lateral collaboration between bots (offering services), decentralization vs. hierarchical control approach and also business considerations. Business integration (and in general, higher-level integration) of the factory cell is done via service-orientation. Business needs are expressed by the production planning and management of the factory cells by monitoring their work status (via specific series), disabling/enabling several routing paths of production, etc.

During operation, reconfiguration may also be needed. For example, a control model for a bot is not anymore valid or production strategies have changed. In these cases, affected bots should be stopped (without paralyzing all the system) and consequently their services would no longer available. During this time, new models can be designed to define the new expected behavior, uploaded to the bots and restart their operation.

5. Conclusions and future work

This paper reports the specified and developed software, under the name of Continuum Development Tools, and provides an overview over the necessary engineering steps to design, analysis and manage automation systems based on service-oriented bots. A special attention was devoted to the framework for the definition of automation bots, which are the main elements in the architecture, after the provided/requested services.

Since the objective is to facilitate the specification and enhance the operation of such systems, based on the already solid foundation introduced in this work, further efforts are required. In the future work can be included the enhancement of the actual status of development enriched with new features and more rich environment (different types of bots and their modules). Important is also the adaptation of advance interaction patterns for service-orientation to increase

the system response and autonomy. In practical terms, evaluation has to be done on the actual case study system by the parameters of performance, design and maintenance efforts, flexibility, and capacity, besides others. Based on one of the requirements, compatibility to the standards of automation (such as IEC 61131 and IEC 61499) must be reached to provide smooth transitions to the new approach.

6. Acknowledgements

The authors would like to thank the European Commission and the partners of the EU IST FP6 project "Service-Oriented Cross-layer infrastructure for Distributed smart Embedded devices" (SOCRADES), the EU FP6 "Network of Excellence for Innovative Production Machines and Systems" (I*PROMS), and the European ICT FP7 project "Cooperating Objects Network of Excellence" (CONET) for their support.

7. References

[1] A.R. Clapham, T.G. Tutin, and D.M. Moore, *Flora of the British Isles*, Edition 3, Cambridge University Press Archive, 1990.

[2] S.A. Ali, H. Seifoddini, and H. Sun, "Intelligent Modeling and Simulation of Flexible Assembly Systems", In *Proceedings of the 37th Conference on Winter Simulation*, Orlando, Florida, December 2005, pp. 1350-1358.

[3] M.N. Huhns, and M.P. Singh, "Service-oriented computing: key concepts and principles", In *IEEE Internet Computing*, Vol. 9, No. 1, 2005, pp. 75-81.

[4] F. Jammes, A. Mensch, and H. Smit, "Service-oriented Device Communications Using the Devices Profile for Web Services", In *Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-hoc Computing*, ACM Press, New York, NY, USA, 2005, pp. 1-8.

[5] F. Jammes, and H. Smit, "Service-oriented Paradigms in Industrial Automation", In *IEEE Transactions on Industrial Informatics*, Vol. 1, No. 1, Feb. 2005, pp. 62-70.

[6] F. Ciancetta, B. D'Apice, D. Gallo, and C. Landi, "Plug-n-Play Smart Sensor Based on Web Service", In *IEEE Sensors Journal*, Vol. 7, No. 5, May 2007, pp. 882-889.

[7] L. Baresi, R. Heckel, S. Thöne, and D. Varró, "Modeling and Validation of Service-Oriented Architectures: Application vs. Style", In *Proceedings of the 9th European Software Engineering Conference held jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM Press, New York, NY, USA, 2003, pp. 68-77.

[8] W.T. Tsai, "Service-Oriented System Engineering: A New Paradigm", In *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering*, IEEE Computer Society, October 2005, pp. 3-6.

[9] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea, "Web Services Engineering: Promises and Challenges", In *Proceedings of the 24th International Conference on Software Engineering*, ACM Press, New York, NY, USA, 2002, pp. 647-648.

[10] A.W. Colombo, and R. Harrison, "Modular and Collaborative Automation: Achieving Manufacturing Flexibility and Reconfigurability", In *International Journal of Manufacturing Technology and Management 2008*, Vol. 14, No. 3/4, pp. 249-265.

[11] F. Depeisses, "Requirements Specification for SODA", Final Version, March 2007, http://www.soda-itea.org.

[12] P. Spieß, and S. Karnouskos, "Maximizing the Business Value of Networked Embedded Systems through Process-Level Integration into Enterprise Software", In *Proceedings of the Second International Conference on Pervasive Computing and Applications*, July 2007, pp. 536-541.

[13] P. Phaithoonbuathong, T. Kirkham, C.S. Mcleod, M. Capers, R. Harrison, and R.P. Monfared, "Adding Factory Floor Automation to Digital Ecosystems; Tools, Technology and Transformation", In *Proceedings of the 2nd IEEE International Conference on Digital Ecosystems and Technologies*, Feb. 2008, pp. 288-293.

[14] J.M. Mendes, P. Leitão, A.W. Colombo, and F. Restivo, "Service-Oriented Process Control using High-Level Petri Nets", *In Proceedings of the 6th IEEE International Conference on Industrial Informatics*, Daejeon (South Korea), July 2008, pp. 750-755.

[15] T. Murata, "Petri nets: Properties, Analysis and Applications", In *Proceedings of the IEEE*, Vol. 77, 1989, pp. 541-580.

[16] D.J. Barrett, L.A. Clarke, P.L. Tarr, and A.E. Wise, "A Framework for Event-based Software Integration", In *ACM Transaction on Software Engineering Methodologies*, Vol. 5, No. 4, 1996, pp. 378-421.

[17] J. Blanchette, and M. Summerfield, *C++ GUI Programming with Qt 4*, Prentice Hall (in association with Trolltech Press), 2006.

[18] H.E. Bal, J.G. Steiner, and A.S. Tanenbaum, "Programming Languages for Distributed Computing Systems", In *ACM Computing Surveys (CSUR)*, Vol. 21, No. 3, ACM Press, New York, NY, USA, 1989, pp. 261-322.

[19] C.A. Petri, *Kommunikation mit Automaten*, Doctoral Thesis, Bonn Institut fuer lnstrumentelle Mathematik, Schriften des IIM, Nr. 3, 1962.

[20] J.L. Peterson, "Petri Nets", In *ACM Computing Surveys*, Vol. 9, No. 3, 1977, pp. 223-252.

[21] Z. Suraj, B. Fryc, Z. Matusiewicz, and K. Pancerz, "A Petri Net System - an Overview", In *Fundamenta Informaticae*, Vol. 71, No. 1, IOS Press Amsterdam, The Netherlands, January 2006, pp. 101 – 119.